60480400

CONTROL DATA
CORPORATION

NETWORK PRODUCTS

NETWORK ACCESS METHOD VERSION 1 FORTRAN APPLICATION PROGRAMMER'S SYSTEM BULLETIN

CDC® OPERATING SYSTEMS:
NOS 1

REVISION RECORD

REVISION	DESCRIPTION
A	Original release at PSR level 477.
(08-15-78)	
Publication No. 60480400	

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning
this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
215 MOFFETT PARK DRIVE
SUNNYVALE, CALIFORNIA 94086

or use Comment Sheet in the
back of this manual

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Revision	Page	Revision	Page	Revision
Cover	–				
Title Page	–				
ii	A				
iii/iv	A				
v	A				
vi	A				
vii	A				
viii	A				
1-1	A				
1-2	A				
2-1 thru 2-6	A				
3-1 thru 3-10	A				
4-1 thru 4-10	A				
5-1 thru 5-6	A				
6-1 thru 6-16	A				
A-1 thru A-6	A				
B-1	A				
B-2	A				
C-1	A				
D-1	A				
Index-1	A				
Index-2	A				
Comment Sheet	A				
Mailer	A				
Back Cover	–				

PREFACE

This Programmer's System Bulletin explains how a user application program should be written to communicate with the network through the Network Access Method (NAM).

Such an application program can be written in one of the high-level languages, such as FORTRAN, COBOL, or SYMPL, where communication is with the various NAM components through interface routines; or it can be written in COMPASS, where communication is provided through a set of macros.

This bulletin does not contain all details concerning the use of NAM; however, it explains the basic principles to the novice user. For more detail concerning the use of NAM, refer to the NAM 1 reference manual.

Examples and routine call formats are presented in FORTRAN, and only the interactive features are discussed. Shading is used on sample listings to highlight coding changes to an earlier example that implement additional features. Shading does not indicate ANSI or non-ANSI usages.

RELATED MANUALS

The NAM applications programmer can find additional pertinent information in the following Control Data Corporation manuals:

Publication	Publication Number
NOS 1 Reference Manual (Volume I)	60435400
NOS 1 Reference Manual (Volume II)	60445300
NOS 1 System Maintenance Reference Manual	60455380
NOS 1 Operator's Guide	60435600
Transaction Facility Version 1 Reference Manual	60455340
Interactive Facility Version 1 Reference Manual	60455250
COMPASS Version 3 Reference Manual	60492600
FORTRAN Extended Version 4 Reference Manual	60497800
Network Products Stimulator Version 1 Reference Manual	60480500
Network Products Network Access Method Version 1 Reference Manual	60499500
Network Products Network Access Method Version 1 Network Definition Language Reference Manual	60480000
Network Products Remote Batch Facility Version 1 Reference Manual	60499600
Network Products 255x Series Communications Control Program Version 3 Reference Manual	60471400
8-Bit Subroutines Reference Manual	60495500
Programming Reference Aids	60158600
731-12 Remote Batch Terminal Operating and Programming Guide	82186800
200 User Terminal Operating and Programming Guide	82136000
714-10/20 Remote Terminal Subsystem Operating Guide	82184500
714-10/20 Remote Terminal Subsystem Reference Manual	82184600

711-10 CRT Display Terminal Operator's Guide	62034100
711-10 CRT Display Terminal Reference Manual	62022700
750 Terminal Subsystem Operator's Guide	62951400
750 Terminal Subsystem Reference Manual	62962800
713-10 Conversational Display Terminal Operator's Guide	62037900
713-10 Conversational Display Terminal Reference Manual	62033400
734 Batch Terminal Operator's Guide	62971500
734 Batch Terminal Reference Manual	62971300
Mode 4C Data Communication Control Procedure System Standard	CDC-STD 1.10.020

CDC manuals can be ordered from Control Data Literature and Distribution Services, 8001 East Bloomington Freeway, Minneapolis, Minnesota 55420.

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLES

INTRODUCTION 1

WHAT IS NAM?

Network Access Method (NAM) is a data message-switching and routing system. It enables host applications to share access with a network of terminals and other host applications.

NAM is not only an interface between programs running in a host and the network, but is also an access method. NAM has its own unique communication protocols. Application programs requiring network access must be specially written.

This bulletin is oriented toward application programs that support interactive terminals. An application to drive a remote batch station, which contains card readers and line printers, can also be written. These features, however, are not discussed in this bulletin.

WHAT DOES NETWORK MEAN?

A network is an interconnected complex that consists of terminals, network processing units (NPUs), couplers, and a host computer (such as the CONTROL DATA® CYBER 170 Models 171, 172, 173, 174 and 175; the CYBER 70 Models 71, 72, 73 and 74; and the 6000 Series Computer Systems). At present, only one host computer is supported. Future networks will provide multihost configurations.

WHY NAM IS NEEDED?

The topology of a network can become extremely complex, consisting of hundreds of different terminals, NPUs, and a host computer. NAM is provided as an interface between application programs executing in the host at one end and the terminals connected to the network at the other end. NAM provides simple, concise, and unique communication protocols, relieving the applications programmer of concern for different and complex protocols needed for such data and message traffic.

NAM also provides the following features:

- Isolation of network communications from the operating system
- Management of network protocol
- Dynamic establishment, maintenance, and termination of data paths between terminals and programs, or between programs
- Buffering and queuing of data for regulation of data flow
- Support of a wide variety of terminals through normalization of data formats (code conversion), as well as the ability to handle transparent unnormalized and unconverted data
- Communication of any program with a number of terminals, addressed individually or as members of a group
- Detection of inactivity on any data path

WHAT ARE THE BASIC NAM COMPONENTS?

The basic NAM components are shown in figure 1-1. A description of these basic components is as follows:

- The Network Interface Program (NIP) – a central processor program running in a dedicated system control point. NIP can read and write into the field lengths of other control points.
- The Peripheral Interface Program (PIP) – the NPU-NAM interface running in a dedicated peripheral processing unit (PPU) for the NIP control point. One or more copies of PIP can reside in the PPUs, depending on the configuration of enabled network nodes.
- The Application Interface Program (AIP) – a collection of relocatable subroutines residing in the field length of all network application programs. AIP performs access control and buffering functions for the application.
- The Communications Control Program (CCP) – a program to drive the terminals connected to the NPUs. It resides in the NPUs and contains TIPs (Terminal Interface Programs). CCP is technically not a part of NAM but must be used when NAM is used.

Other NAM components execute as application programs running at normal control points. The following components are not dedicated and can be rolled out:

- The Network Supervisor (NS) – coordinates the activities of the various NPUs. NS is responsible for loading the software into the NPUs, and for establishing and controlling physical paths through the communication network.
- The Communication Supervisor (CS) – coordinates the line-oriented and terminal-oriented activities of the host computer.
- The Network Validation Facility (NVF) – is responsible for granting network access to the terminal user.

The above components enable any user to write his own application. Various standard application programs provide a set of commonly used communications functions:

- The Remote Batch Facility (RBF) – supports remote job entry terminals, such as 200 User Terminals or HASP workstations.
- The Interactive Facility (IAF) – supports interactive control statement processing.
- The Transaction Access Facility (TAF) – supports transactional terminal operation.
- The Terminal Verification Facility (TVF) – provides terminal verification tests.

HOW IS NAM USED?

The main user of NAM is a programmer who writes installation applications. Application programs communicating with terminals, or with other application programs, must call the appropriate AIP routines. The AIP routines interface between the application and other NAM components.

Access to the network must be initialized by calling the AIP NETON routine. After NETON has been granted by NAM, other AIP routines can be called. These routines enable data and supervisory message traffic between the application and the network. When an application no longer requires use of the network, it must call the AIP NETOFF routine, which causes NAM to disconnect the application from the network.

In the following sections, some basic terms and a few FORTRAN-written application programs are presented. The programs start with a simple program showing connection-establishment steps. The programs are expanded later to a multiterminal conversation program. Parallel mode and application-to-application communication program examples are also presented.

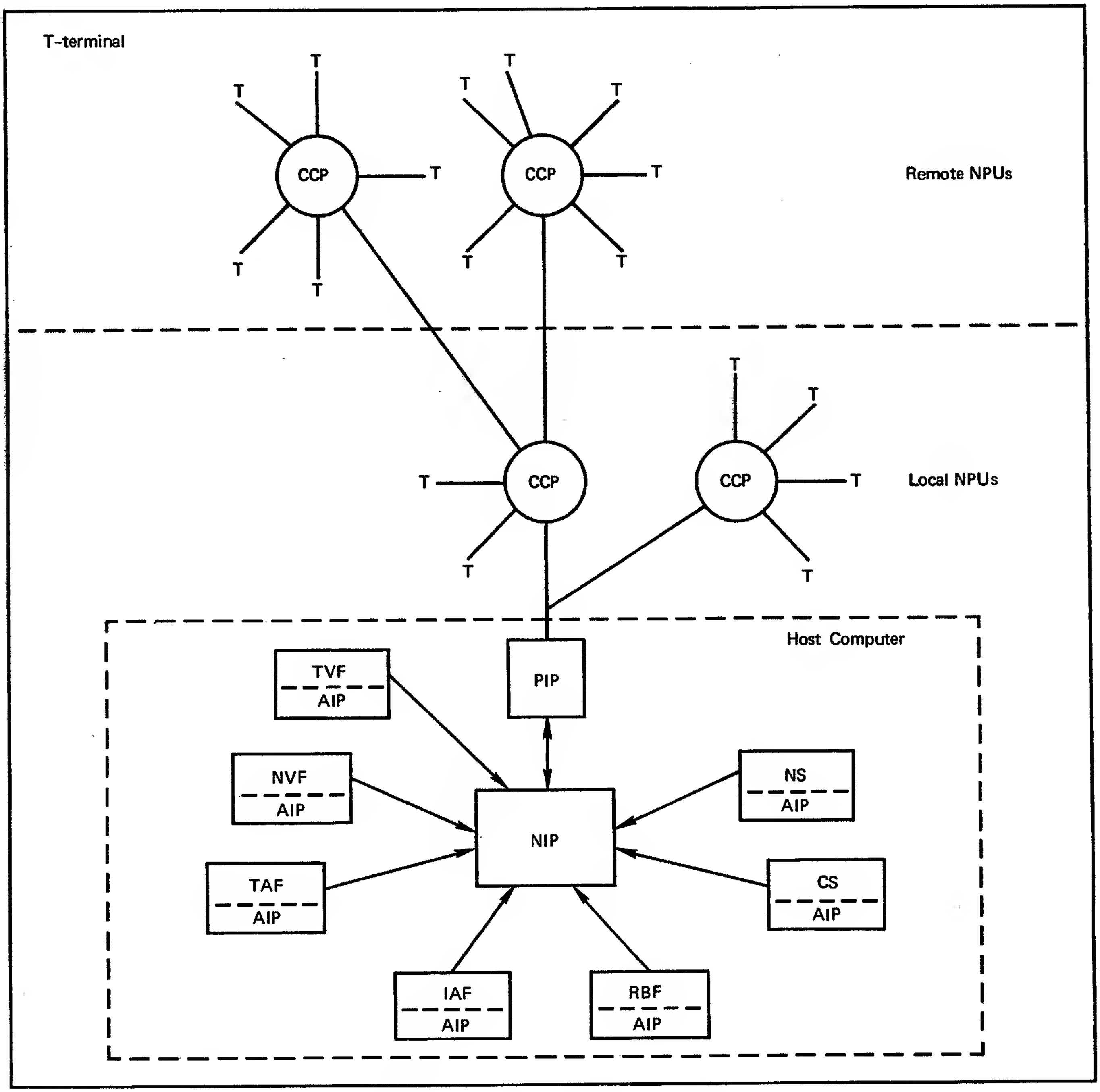


Figure 1-1. Network Software Relationships

Before writing our first application program, some basic terms are essential to further our understanding of NAM.

THE APPLICATION CONNECTION NUMBER

When data passes between a terminal and an application, a message path exists between the two. This message path is called a logical connection. After access, a terminal is logically connected to one application at a time. It can be switched from application to application as needed. Applications can be connected simultaneously to many terminals.

This logical connection is identified as the application connection number (ACN). The ACN is a 12-bit integer value, assigned by NAM, used to specify a particular path. An ACN that becomes available because of disconnections is reassigned to a subsequent connection. An ACN of 0 indicates the control connection along which certain supervisory messages are sent and received.

When the application inputs data from a specific terminal or another application, it calls the AIP NETGET routine, and specifies the ACN from which input is requested.

When the application program outputs data to a specific terminal or another application, it calls the AIP NETPUT routine, again specifying the ACN to which output is to be sent.

Notice that ACNs and connections are not the same; while connections are always unique at a given time, ACNs are unique only within the same application program. For example, in figure 2-1, an ACN of 2 identifies connection b for application A, connection e for application B, and connection h for application C.

Also notice that application-to-application connections are identified by two ACNs, which are not necessarily the same number. For example, application A is assigned an ACN of 5 for connection g, while application C is assigned an ACN of 1 for the same connection.

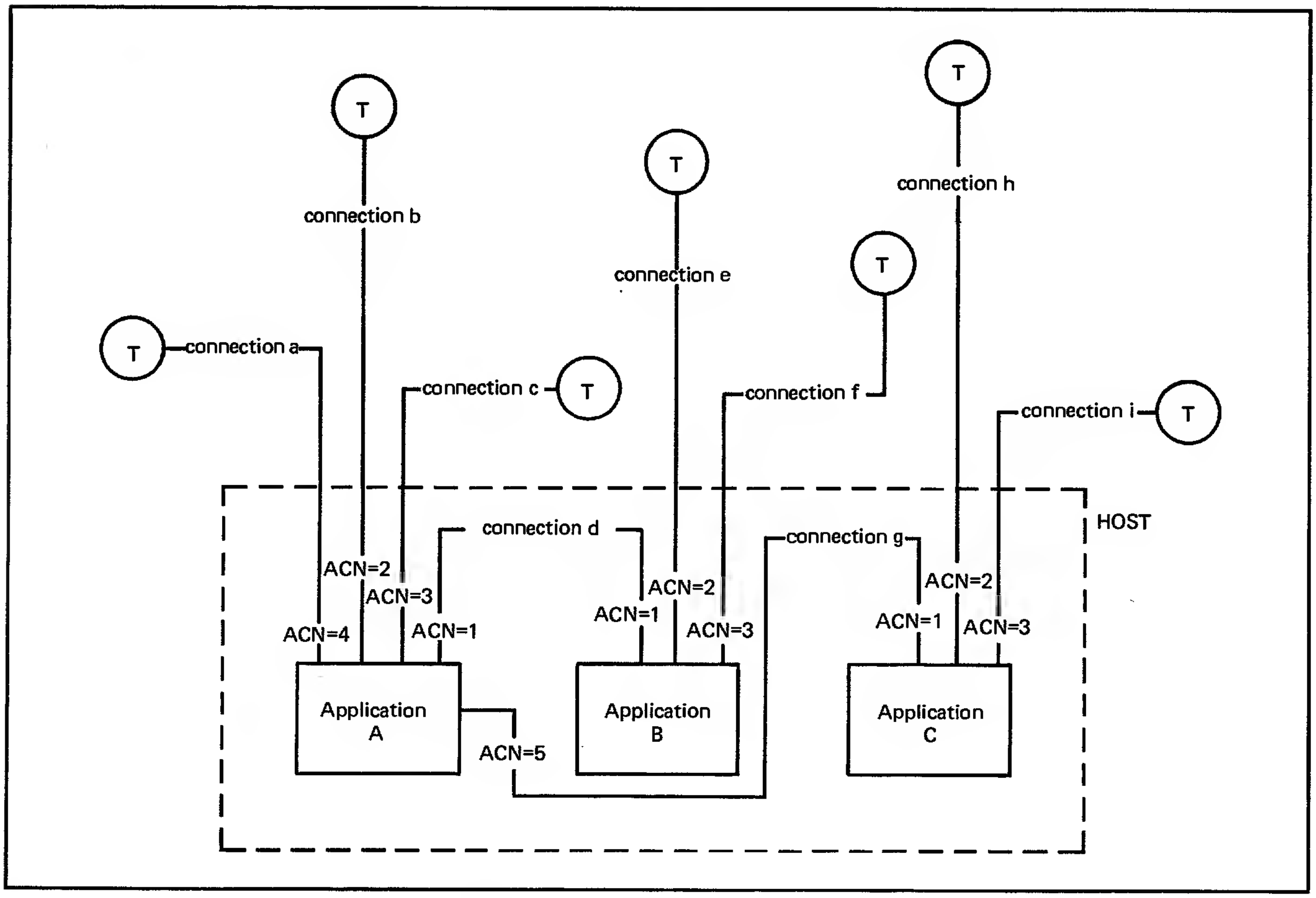


Figure 2-1. Application Connection Numbers

THE MESSAGE

A message is a logical unit of information exchanged between NAM and the application. It can be a line of data, a screen, or a file. A message can be of arbitrary length and content. If long, the message is broken into smaller units called blocks.

THE BLOCK

A block is the basic unit of information exchanged between NAM and the application. Block length is dependent on the terminal device, but cannot exceed 410 60-bit words or 2043 characters.

There are two types of blocks involved in data transfer:

BLK blocks – the first n-1 blocks of the message when a message is composed of n blocks.

MSG block – the last or nth block of a message. See figure 2-2.

A message can contain a MSG type block only when the message is short enough to be contained in one block.

In addition to BLK blocks and MSG blocks, there are two special type blocks:

Null blocks – blocks containing no physical information, but rather indicate a status. A status results when NAM has no input for the application input operation.

SM blocks – supervisory message blocks. They are not part of a message. They convey information about the data and connections. Supervisory messages consist of one block and are used for control purposes.

SUPERVISORY MESSAGES

Application programs exchange supervisory messages (SM) with NAM to control logical connections and data flow. There are two types of supervisory messages:

Asynchronous supervisory message (ASM)

Synchronous supervisory message (SSM)

All supervisory messages that are sent or received on an ACN of 0 are asynchronous, and are exchanged independent of the data; conversely, supervisory messages that are sent or received on an ACN not equal to 0 are synchronous, and are exchanged at the same time as the data. Synchronous supervisory messages can be used as markers in the data stream. All supervisory messages have a special role and format, and are listed in appendix C.

THE TEXT AREA

The text area (ta) is a buffer used to exchange information between NAM and the application program. See figure 2-3. Text area length cannot exceed 410 central memory words, and can contain one of the following:

- A BLK type block
- A MSG type block
- An SM type block

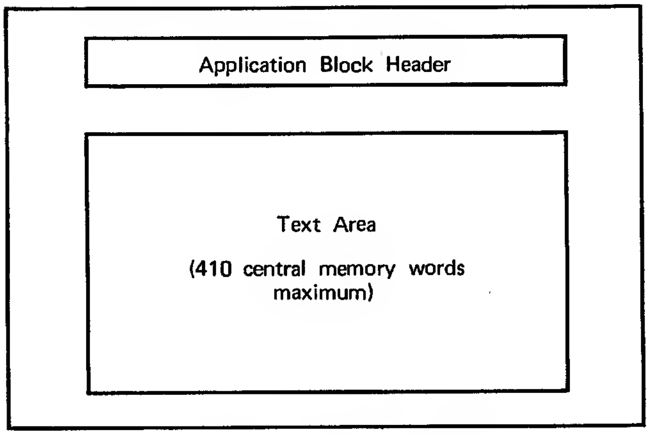


Figure 2-3. Application Block Header and Text Area

THE APPLICATION BLOCK HEADER

Every block or supervisory message passed between NAM and the application must be accompanied by one word which is called the application block header (ABH). See figures 2-3 and 2-4. Each header contains detailed information describing the text area information it accompanies, indicating what NAM or the application should do with it.

On output, the application creates the ABH and NAM interprets it. On input, NAM creates ABH and the application interprets it. The ABH and the text area can reside in two different areas in memory; therefore, they need not be contiguous. Usages of the ABH and text area are shown in the following examples.

After inputting the ASCII characters

THIS IS INPUT

the contents of the ABH word are configured, as shown in figure 2-5. The contents of the text area are shown in figure 2-6.

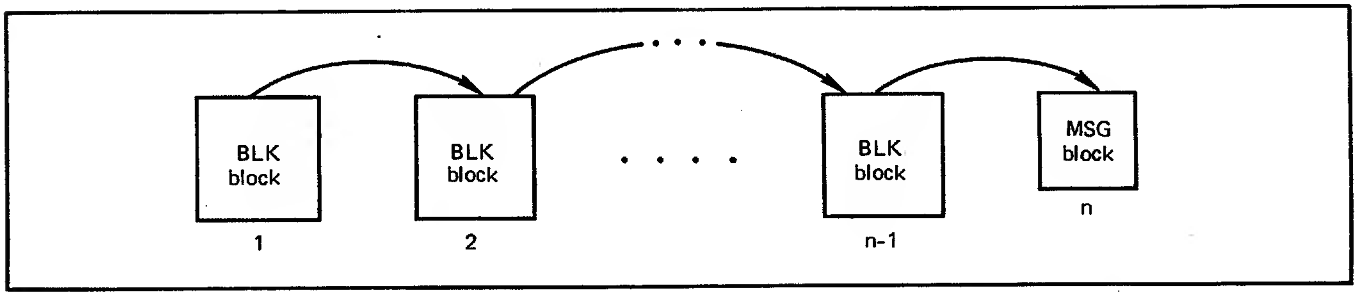


Figure 2-2. BLK and MSG Message Blocks

59	53	41	23	19	11 0
ABT	ADR	ABN	ACT	FLAGS	TLC

ABT Application block type:

0 = Null block, input only.

1 = BLK type block, first n-1 blocks of a message.

2 = MSG type block, which is the last block of a message, or the only block.

3 = Supervisory message, input or output.

ADR Addressing information; contains the ACN.

ABN Application block number; a number created by the application to keep track of transferred blocks. It can be:

A serial number

The block's core address

The block's disk address.

An external name

or whatever the application chooses appropriate to identify the block.

ACT Application character type. This is a number identifying the associated block character type and can be:

1 = 60-bit words. Must be used for application-to-application connections and for supervisory messages when an ADR is not equal to 0. Cannot be used for application-to-terminal connections.

2 = 8-bit ASCII characters 7.5 characters per word, used for application-to-terminal connections, and for synchronous supervisory messages when ADR is equal to 0. See section 2.

3 =·8-bit ASCII characters, right-justified in 12-bit bytes 5 per central memory word. Used as an alternate form for application-to-terminal connections. Cannot be used for supervisory messages.

4 = 6-bit display code characters, 10 per word.

Alternate form for application-to-terminal connections.

Cannot be used for supervisory messages.

FLAGS Various flags. Their use is explained in section 5.

TLC Text length in units specified by the ACT.

Figure 2-4. Application Block Header Format

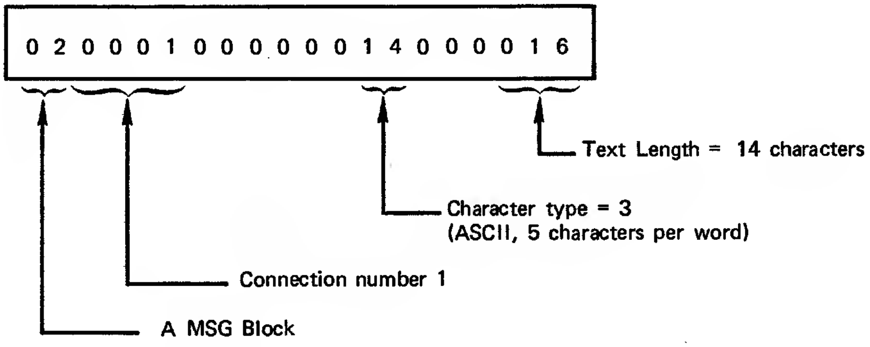


Figure 2-5. ABH Configuration of 'ΔTHISΔISΔINPUT'

Δ	T	H	I	S
Δ	I	S	Δ	I
N	P	U	T	

Figure 2-6. Content of Text Area of 'ΔTHISΔISΔINPUT'

Upon outputting the 18-character display code message

ENTER INPUT PLEASE

the contents of the ABH word are configured, as shown in figure 2-7. The contents of the text area are shown in figure 2-8.

BASIC PROTOCOLS INVOLVING SUPERVISORY MESSAGES

A protocol is the ordered step-by-step process of exchanging messages. An example of a supervisory message protocol is the pair of messages for a request for some action to be performed and a response to it (either acceptance or rejection). As mentioned earlier, supervisory messages can be initiated by NAM and responded to by the application, or vice versa.

Every supervisory message is identified by two function codes:

Primary function code (PFC)

Secondary function code (SFC)

All available PFCs and SFCs have numerical values assigned to them. Each is also represented by a symbolic name which can be gained through the use of keywords. (See appendix B.)

A specific supervisory message is indicated by the pair PFC/SFC. This pair forms unique function requests. For example, the primary function code CON together with the secondary function code REQ forms the unique function code CON/REQ (connection request).

E	N	T	E	R	Δ	I	N	P	U
T	Δ	P	L	E	A	S	E	00	00

Figure 2-8. Content of Text Area of 'ENTERΔINPUTΔPLEASE'

Along with every PFC and SFC, there are three other sections of a supervisory message:

EB — A 1-bit field error bit which is set to 1 to indicate error or abnormal response, either from NAM to the application or vice versa.

RB — A 1-bit field response bit which is set to 1 in every normal response to a previous supervisory message, either set by the application as a response to NAM or vice versa.

Parameter Field — 1 to 63 words of parameters passed between NAM and the application, dependent on the specific request.

The general format of a supervisory message is shown in figure 2-9. The ABH accompanies every block being transferred whether it is a supervisory message block or a data block. The ABH format for an asynchronous supervisory message is shown in figure 2-10. The ABH for a synchronous supervisory message is shown in figure 2-11.

NAM TO APPLICATION SUPERVISORY MESSAGES

PFC/SFC	Meaning
CON/REQ	NAM informs the application that a terminal or another application requests a logical connection.
CON/CB	Logical connection is broken (for example, line disconnect).
FC/INIT	NAM informs the application of the logical initialization of a connection.
FC/ACK	NAM acknowledges the delivery of a previously submitted block by the application.

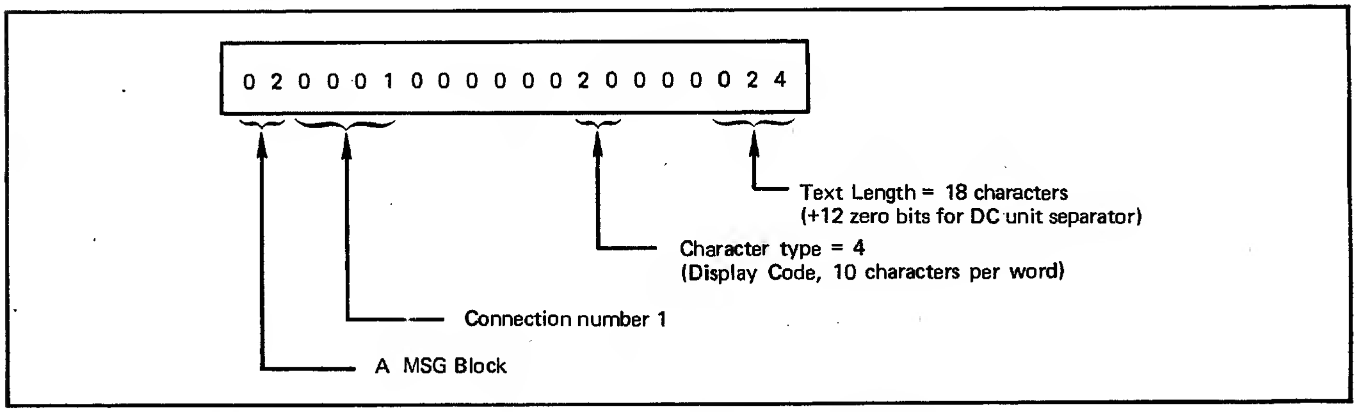

0 2 0 0 0 1 0 0 0 0 0 0 2 0 0 0 0 0 2 4
Text Length = 18 characters
(+12 zero bits for DC unit separator)
Character type = 4
(Display Code, 10 characters per word)
Connection number 1
A MSG Block

Figure 2-7. ABH Configuration of 'ENTERΔINPUTΔPLEASE'

NAM to application messages are also used for the following functions:

- Suspension or resumption of data traffic
- Detection of an inactive connection
- Notification of network shutdown
- Logical error in response to an illegally formatted request previously submitted by the application

For the complete list of NAM to application supervisory messages, refer to appendix C.

APPLICATION TO NAM SUPERVISORY MESSAGES

<u>PFC/SFC</u>   <u>Meaning</u>

CON/ACRQ   The application requests connection to another application. Application can initlate connection request to another application, but cannot do the same to a terminal because this connection request comes only from NAM.

CON/END   The application informs NAM that it has completed all processing on a logical connection.

MSG/LOP   The application sends a message to the local operator.

Application to NAM messages are also used for the following functions:

- Temporarily switching OFF and ON data input transmission
- Definition of terminal characteristics
- Changing input character type as chosen by the application, in contrast to what is previously transferred by NAM

For the complete list of application to NAM supervisory messages, refer to appendix C.

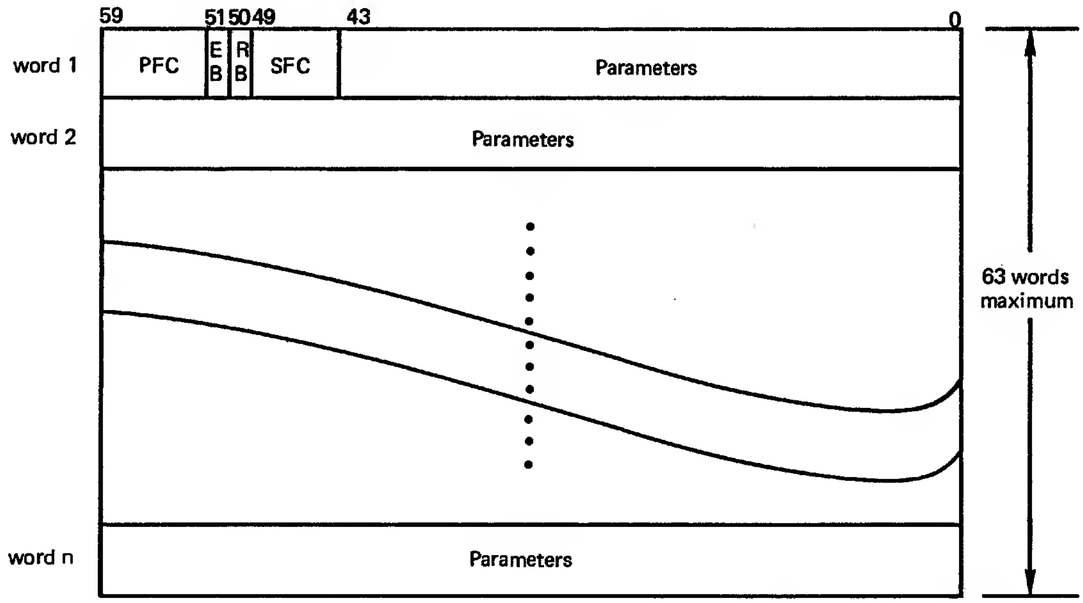


Figure 2-9. Format of Supervisory Message

symbolic name	ABT	ADR		ACT		
value	3	0		1		

Figure 2-10. ABH for an Asynchronous Supervisory Message

symbolic name	ABT	ADR		ACT		
value	3	≠0		2		

Figure 2-11. ABH for a Synchronous Supervisory Message

When writing an application in one of the high-level languages, AIP routines must be called in order to gain network access.

All AIP routines begin with NET; routines used internally by AIP begin with NP$. To avoid possible naming conflicts, variables and entry point names should not begin with NET or NP$.

AIP routines are divided into three groups:

- Informative and control routines

NETON	Establishes network access.
NETOFF	Ends network access.
NETWAIT	Temporarily suspends the application.
NETDBG	Turns NAM debugging option on or off.
NETSTC	Turns NAM statistics capability on or off.

- Data transfer routines

Input routines (from NAM to the application):

NETGET	Gets input on a specified connection.
NETGETL	Gets input on a connection that is a member of a list.
NETGETF	Gets input on a specified connection, into a fragmented buffer.
NETGTFL	Gets input on a connection that is a member of a list, into a fragmented buffer.

Output routines (from the application to NAM):

NETPUT	Sends output to a specified connection.
NETPUTF	Sends output on a specified connection, from a fragmented buffer.

- Parallel mode control routines

NETSETP	Selects or terminates parallel mode.
NETCHEK	Checks for completion of NAM call while in parallel mode.

The preceding features of NAM, and their communication protocols, are expanded upon in the following subsections.

NETON AND NETOFF

In order to access the network and establish connections, the application must first call NETON. The FORTRAN format is

```
CALL NETON (nHaname,nsup,status,minacn,maxacn)
```

NETON parameters are as follows:

aname Application name, one to seven alphanumeric display code characters, left-justified with blank fill. This name is used to identify the application, and must be made known to NAM before using it.

nsup A single-word variable into which NAM stores various flags during subsequent network calls. See figure 3-1.

status Another single variable, which indicates NETON success or failure. The values are:

0 NETON is successful.

1 Rejected because the network is not available.

2 Rejected because of duplicate application name.

3 Rejected because application name is not known to NAM.

minacn A number that indicates the smallest ACN the application accepts.

maxacn A number that indicates the largest ACN the application accepts.

NAM assigns ACNs starting at minacn and not exceeding maxacn. The minacn and maxacn parameters must conform to the following specifications:

$0 < \text{minacn} \le \text{maxacn} \le 4095.$

Example:

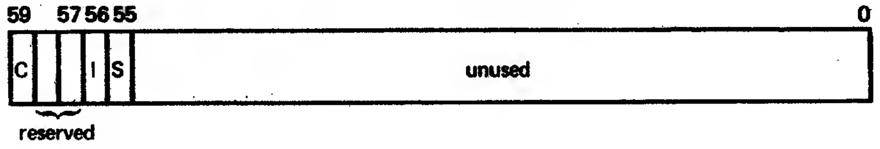
```
CALL NETON (SHMYAPP,NSP,NST,5,12)
```

Application MYAPP requests network access and is willing to accept only eight connections, starting with connection number 5 (minacn) and not exceeding connection number 12 (maxacn). C, I, and S bits are returned in nsup (NSP) and accept/reject status is returned in status (NST).

When the application desires to terminate network access, it calls NETOFF (no parameters are required):

```
CALL NETOFF
```

After calling NETOFF, the application continues to execute under control of the operating system. In order to resume network access, NETON is called again.



C Complete bit. This bit is applicable in the parallel mode of operation only. Refer to section 6 for further explanation about this bit.

I Input available bit. Set to 1 when input other than an asynchronous supervisory message is available to the application. Set to 0 otherwise. This bit is set only after calling NETWAIT.

S Asynchronous supervisory message bit. Set to 1 if there are asynchronous supervisory messages available to the application. Set to 0 otherwise. In contrast with the I bit, this bit is usually set or cleared after each AIP call, except NETSETP. (Refer to section 6.)

Figure 3-1. NSUP Word

NETGET, NETPUT, AND NETWAIT

An application program can input data (NETGET), output data (NETPUT), and suspend processing (NETWAIT). These options are permitted by AIP statements. They are discussed in the following subsections.

NETGET

After network access is made possible, supervisory messages and input/output data traffic can begin. This can be accomplished by calling NETGET for input and NETPUT for output.

The FORTRAN format of NETGET is

```
CALL NETGET (acn,ha,ta,tlmax)
```

This routine inputs one data block or a supervisory message from the specified connection. The header of the block is placed in the header area (ha), and the body of the block in the text area (ta). Text area is an array of at least maximum text length (tlmax) words.

An application connection number of 0 is used to obtain an asynchronous supervisory message. If no blocks are available for the specified connection, a null block is returned to the application, an ABH with an ABT of 0 is placed in the header area, and the text area remains unchanged. If the block is longer than the maximum text length, the block is not transmitted, but remains queued within NAM. An ABH, which contains an IBU bit set to one and the true length of the block, is returned to the application. (See figure 3-2. See also section 5.) The text area (ta) remains unchanged.

Example:

```
CALL NETGET (0,IHA,ITA,63)
```

This routine requests input from an ACN of 0. The ABH is placed in IHA, and the text is placed in ITA. ITA is an array of at least 63 central memory words.

NETPUT

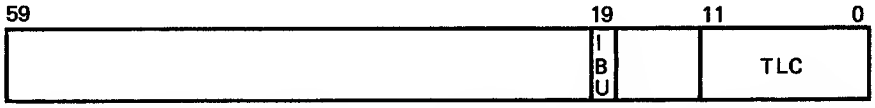
The second routine is NETPUT. Its calling format is

```
CALL NETPUT (ha,ta)
```

This routine outputs one data block or a supervisory message from the text area (ta) to the connection; it does so according to the information placed in the header area (ha). In contrast to NETGET, an ABH is constructed and placed in the header area (ha) prior to calling NETPUT.

The header area must at least contain the following information:

- The connection number (placed in the ADR field)
- The block type (placed in the ABT field)



IBU bit Set to 1 when the block's length is longer than the tlmax specified in NETGET (also applicable to NETGETL, NETGETF and NETGTFL).

TLC Contains block's actual length.

Figure 3-2. Format for Data Block Header

- The block character type (placed in the ACT field)
- The block length in units (placed in the TLC field)

For the following example, assume that NAM has assigned an ACN of 5. To output the display code characters

INPUT PLS

to the ACN of 5, an ABH is constructed which contains the information shown in figure 3-3.

The header area is given an octal value by the statement

IHA = 0200050000000200000012B

The following steps are required to prepare the text area and send the message:

ITA(1) = 10H INPUT PLS

ITA(2) = 0

CALL NETPUT (IHA,ITA)

NETWAIT

If the application is suspended, or when the NSUP word should be updated, the NETWAIT subroutine must be called:

CALL NETWAIT (time,flag)

NETWAIT parameters are as follows:

time Time in seconds that the application is suspended; must be in the range 1 through 4095. If smaller than 1, the default value of 1 is substituted. If greater than 4095, the default of 4095 is substituted.

flag A single variable that indicates recall condition. The values are:

0 Returns control when input is available or time specified has elapsed, whichever occurs first.

1 Returns control only when time specified has elapsed, regardless of input availability.

FIRST STEPS IN CONNECTION ESTABLISHMENT

Suppose MYAPP, which handles up to a maximum of 10 terminals, is a configured application known to the network software. NETON is called by writing

CALL NETON (SHMYAPP,NSUP,NSTAT,1,10)

Network access becomes possible only after NETON is complete with NSUP set to 0. NAM responds with CON/REQ supervisory messages on behalf of every terminal or application.

Request by request, the application must decide whether to accept or reject the CON/REQ supervisory message. Rejection can occur for whatever reason the application chooses; for example, when the application deals with TTY-compatible terminals only and some other device requests a connection.

If the application rejects the CON/REQ, it sets the error bit to 1 and NETPUTs this rejection message to NAM. If the application accepts the CON/REQ, it sets the response bit to 1 and NETPUTs this response message to NAM. At this point, data transfer between the requesting terminal and the application is not yet possible. The application can continue with some other processing or wait.

After NAM receives the CON/REQ, and the response bit is 1, NAM responds with a flow-control initialized (FC/INIT) supervisory message and the application responds by accepting it. Data traffic now begins between the application and the requesting terminal or another application.

The connection establishment steps are summarized in table 3-1. In this table, a specific supervisory message is indicated by the PFC/SFC pair.

A response to a supervisory message is indicated by the triplet

PFC/SFC/OPT

where:

PFC/SFC are the supervisory message identifiers.

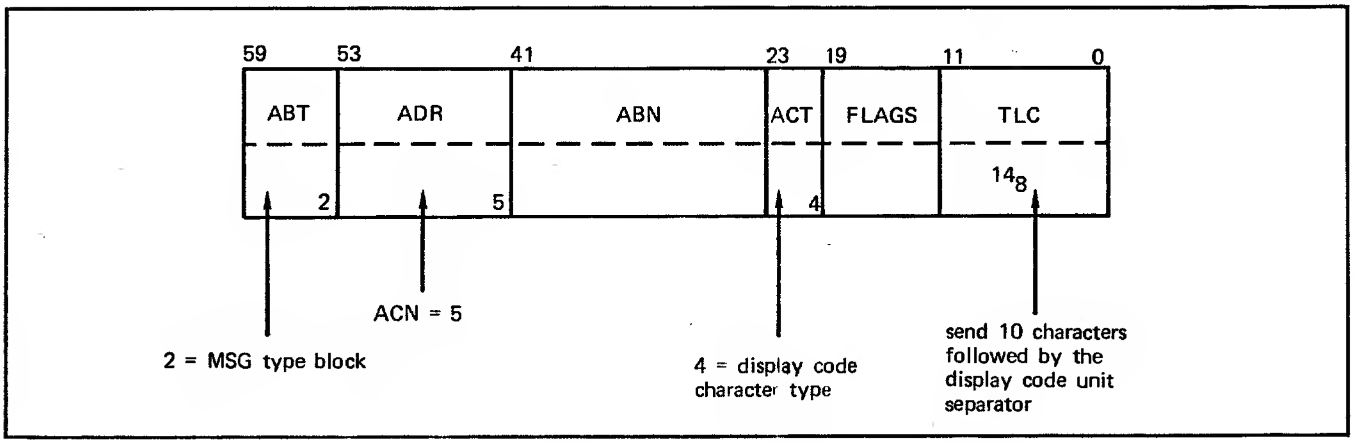

59
53
41
23
19
11
0
ABT
ADR
ABN
ACT
FLAGS
TLC
2
5
4
14_8
ACN = 5
2 = MSG type block
4 = display code character type
send 10 characters followed by the display code unit separator

Figure 3-3. ABH Configuration of 'ΔINPUTΔPLS'

TABLE 3-1. CONNECTION ESTABLISHMENT CHART

Step No.	Description of Action	Application AIP Routine Called	Message Flow	NAM	Terminal (or Application)
1.	Application calls NETON.	NETON	→		
2.	NETON was accepted by NAM (status of 0).		←		
3.	A terminal user, or another application, is requesting the application; meanwhile, the application can drop the CPU and wait for supervisory message.	NETWAIT			←
4.	NAM sends a CON/REQ supervisory message; application receives the message only after calling NETGET on an ACN of 0.	NETGET	←	CON/REQ	
5.	Application accepts the CON/REQ by setting RB=1; N stands for RB=1, which is a normal response, and NETPUTs this message to NAM.	NETPUT	→	CON/REQ/N	
6.	Application can now do some other computations or relinquish the CPU by calling NETWAIT.	NETWAIT			
7.	Flow-control is initialized by NAM sending the FC/INIT.	NETGET	←	FC/INIT	
8.	Application accepts the FC/INIT and NETPUTs the message back, setting RB=1.	NETPUT	→	FC/INIT/N	
9.	Data traffic can start now on the assigned ACN not equal to 0.	NETPUT NETGET	→ ←		

OPT indicates the type of response:

N Normal response (RB=1)

A Abnormal response (EB=1)

For example, CON/REQ/N means a normal response to a CON/REQ supervisory message.

The format of the supervisory messages involved in connection management is shown in figure 3-4.

A normal response of the application to NAM for the CON/REQ SM is shown in figure 3-5. An abnormal response is shown in figure 3-6. FC/INIT from NAM to the application is shown in figure 3-7; a normal response of the application to this request is shown in figure 3-8. An abnormal response to this request is shown in figure 3-9.

FIRST SAMPLE PROGRAM

EASY, a sample program, is shown in figure 3-10. This sample program demonstrates how an application program communicates with the network. It identifies the use and format of the following NAM terms:

- The application connection number (ACN)
- The application block header (ABH)
- The supervisory message (SM)
- The text area (ta)
- The header area (ha)

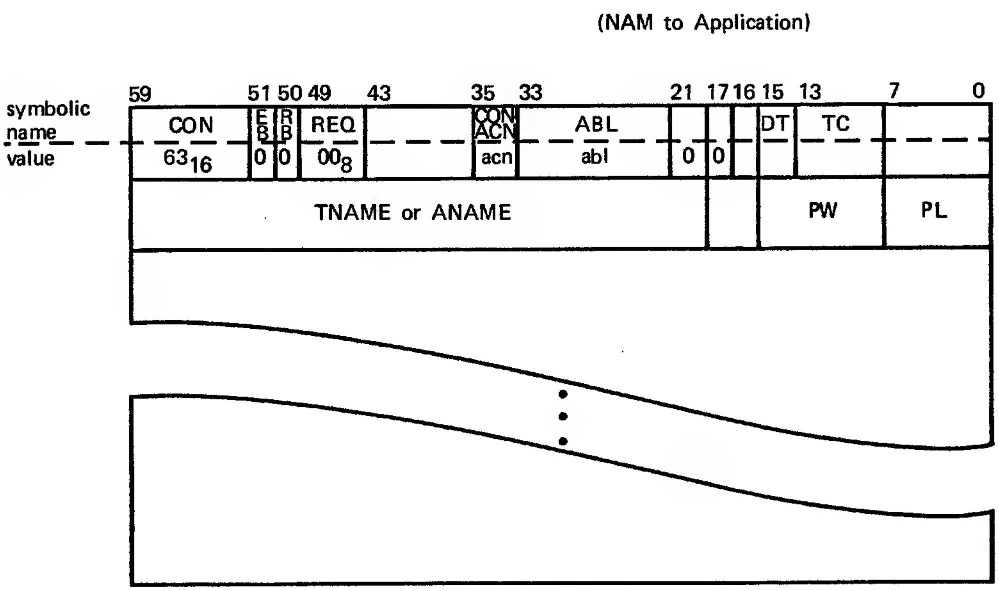
EASY also identifies the use and format of the following AIP routines:

- NETON
- NETWAIT
- NETGET
- NETPUT
- NETOFF

The remarks column, on the right in figure 3-10, explains the step-by-step activity of the program.

NFETCH AND NSTORE SUBROUTINES

Using masks and special purpose constants, accessing supervisory messages, and block header fields is awkward. Besides the disadvantage of having to be aware of the exact structure and value, it is good programming practice not to use masks and special purpose constants. Instead, the NFETCH and the NSTORE subroutines are used. These routines must scan a table for matching the keywords; scanning causes some increase in program execution time. Therefore, it is advisable to set up as many constants as possible, at initialization time only.

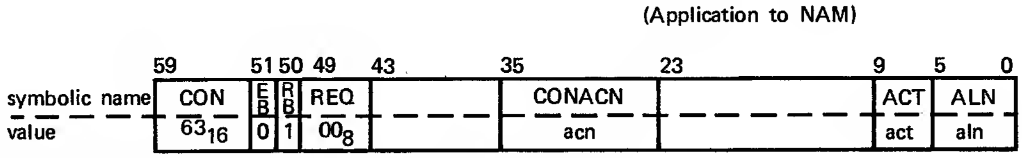


acn The assigned application connection number.

abl The application block limit.

DT, TC, PW, and PL are device type, terminal class, page width and page length, respectively. TNAME or ANAME are 1 through 7 alphanumeric characters of terminal or application name, respectively, left-justified, blank-filled with a leading alphabetic character.

Figure 3-4. Format of Connection Management Supervisory Messages



act The initial input application character type as determined by the application.

aln The application list number.

Figure 3-5. Normal Response Format for Connection Request (CON/REQ/N)

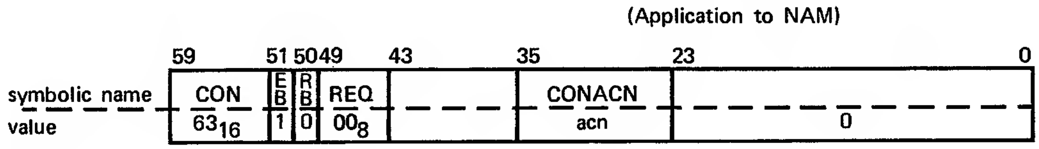


Figure 3-6. Abnormal Response Format for Connection Request (CON/REQ/A)

(Application to NAM)

	59	51	50	49	43	35	23	0
symbolic name	FC	EB	RB	INIT		FCACN		
value	83_{16}	0	0	07_8		acn		

Figure 3-7. Flow-Control/Initialized Format (FC/INIT)

(Application to NAM)

	59	51	50	49	43	35	23	0
symbolic name	FC	EB	RB	INIT		FCACN		
value	83_{16}	0	1	07_8		acn		

Figure 3-8. Normal Response Format for Connection Initialized (FC/INIT/N)

(NAM to Application)

	59	51	50	49	43	35	23	0
symbolic name	FC	EB	RB	INIT		FCACN		
value	83_{16}	1	0	07_8		acn		

Figure 3-9. Abnormal Response Format for Connection Initialized (FC/INIT/A)

NFETCH FUNCTION

NFETCH is used to fetch a specified field. For example:

var = NFETCH (arr,nLkeyword)

where:

arr indicates an array that contains the table from which the specified field is extracted. If an arr is 0, then a constant value represented by keyword is returned rather than a field from a table.

keyword indicates a symbolic field name.

For example:

KKK = NFETCH (TA,6LPFCSFC)

KKK contains the combined primary and secondary function code from array TA.

LLL = NFETCH (0,6LFCINIT)

LLL contains the constant represented by FCINIT (8307_{16}).

NSTORE FUNCTION

NSTORE is used to store a value in a table. For example:

CALL NSTORE (arr,nLkeyword,val)

where:

arr indicates an array that contains the table.

keyword is a symbolic name of the field in the array into which the value is to be stored.

val is a numeric or alphanumeric value to be stored.

```
Step
No.                                                                      Remarks

              PROGRAM EASY (OUTPUT)
             INTEGER PF,SFC,CON,REQ,FC,S,SMHDR                           Set constants
             INTEGER OTHDR,HA,TA(63)                                     $63_{16}$
             CON=30600000000000000000B
             REQ=0
             FC=40600000000000000000B                                    $83_{16}$
             INIT=00003400000000000000B                                  $07_{16}$ (in bits 44 through 49)
             SMHDR=03000000000004000001B                                 ABH for 1 word supervisory message
             DTHDR=02000000000002000024B                                 ABH for 10 character display code
 1           CALL NETON (5HMYAPP,NSUP,NSTAT,1,10)
 2           IF (NSTAT.EQ.0)GO TO 4                                      Check if NETON has been accepted.
             PRINT 100,NSTAT
       100   FORMAT (*NETON FAILED,NSTAT=*,020)                          NETON failed. Print NSTAT value to find out
                                                                         failure reason,disconnect application and drop.
             STOP 111
 3             •                                                         Some other computations can now be made,
               •                                                         followed by a NETWAIT.
               •
       5     CALL NETWAIT(4095,0)
             S=SHIFT(NSUP,-55).AND.1
             I=SHIFT(NSUP,-56).AND.1
             IF(S.EQ.1) GO TO 6
             IF (I.EQ.1)GO TO 10
             GO TO 5
 4       6   CALL NETGET(0,HA,TA,63)                                     Supervisory message is available and can be
                                                                         accessed through a NETGET.
             PFC=TA.AND.77600000000000000000B
             SFC=TA.AND.00037400000000000000B
             IF (PFC.EQ.CON.AND.SFC.EQ.REQ) GO TO 7                      Is it a CON/REQ?
             IF (PFC.EQ.FC.AND.SFC.EQ.INIT) GO TO 8                      Is it an FC/INIT?
             PRINT 101,HA,TA
       101   FORMAT(*UNKNOWN SUP. MESSAGE *,64(/,1X,020/))               Unknown supervisory message, print header
                                                                         and text areas, disconnect and drop.
             CALL NETOFF
             STOP 222
      C      CON/REQ PROCESSING
         7   ACN=SHIFT(TA,-24).AND.7777B                                 Proceed with a normal response to CON/REQ.
             TA=TA.OR.00040000000000000300B                              Set RB=1, and ACT=3.
             HA=SMHDR                                                    Set ABH for 1 word supervisory message.
 5           CALL NETPUT (HA,TA)                                         Send response to NAM.
               •
               •
               •
 6           GO TO 5                                                     Look for input or another supervisory message.
      C      FCINIT PROCESSING
 7       8   ACN=SHIFT(TA,-24).AND.7777B
             TA=TA.OR.00040000000000000000B                              Set RB=1
             HA=SMHDR
 8           CALL NETPUT(HA,TA)
               •                                                         At this point, input data can be available or
               •                                                         output data can be sent to terminal on
               •                                                         assigned ACN.
             TA=10H INPUT PLS                                            Construct the message "INPUT PLS".
             TA(2)=0
             ACN=SHIFT(ACN,42)                                           Set ABH to reflect display data.
             HA=DTHDR.OR.ACN                                             Send the output message and go wait for input.
 9           CALL NETPUT(HA,TA)
             GO TO 5
      C      INPUT IS AVAILABLE
        10   CALL NETGET(ACN,HA,TA,1)                                    Input is available here.
               •
               •
               •
             END
```

Figure 3-10. Sample Program EASY

KEYWORDS

Keywords can be made easier if the logic behind the keyword names is understood. All the application block header word keywords start with ABH. For example:

ABHABT	is the keyword for the ABT field in the application block header word.
ABHACT	is the keyword for the ACT field in the application block header word.

Also for the text area, associated fields of every supervisory message are prefixed with the PFC name. For the PFC=CON, the ACN and ALN fields are referenced as CONACN and CONALN, respectively. For the PFC=FC, the ACN and ABN fields are referenced as FCACN and FCABN, respectively.

For the general usage of a supervisory message, the following keywords exist:

PFC	The primary function code field
SFC	The secondary function code field
PFCSFC	The combined field of the PFC and SFC
EB	The error bit field
RB	The response bit field
RC	The reason code field

For the complete list of all the keywords, refer to appendix B.

The first sample program revised, using NFETCH and NSTORE, is shown in figure 3-11.

THE APPLICATION LIST NUMBER

The application list number is used to avoid specifying different ACNs for every terminal that sends input. The ACN is specified whenever data is input from a specific connection, which is equivalent at any given time to one specific terminal.

On input only, it is possible for the application to group several ACNs together to form the ALN. The ALN is a 6-bit integer value that is determined by the application (in contrast to the ACN, which is assigned by NAM).

The ALN enables the application to receive data from several terminals without specifying the ACNs for every terminal. By specifying the ALN only, data comes in a round-robin fashion, each time from one single terminal.

In order to input data using ALNs, the AIP NETGETL routine is called, rather than NETGET.

When responding to a NETGETL request, ACNs with empty input are skipped, and data is transferred only from the next ACN that has input available. Using a specific ACN (on NETGET) always results in the return of a null block if there is currently no input available from the specified ACN. NETGETL returns the null block only if there is no input available for all the ACNs in the specified ALN.

The ALN is both more convenient and more efficient than ACN. The ALN should be used whenever an application handles all terminals the same, in which case it places them all in the same ALN. If an application desires to handle batch and interactive terminals in a different manner, it assigns all the batch terminals to ALN=1, and all the interactive terminals to ALN=2.

The ALN is made known to NAM by specifying it in every response of the application to the CON/REQ message from NAM. Recalling the normal response format of the application to CON/REQ in figure 3-5, the ALN field is shown in figure 3-12. In the sample revised program, to access all terminals on an ALN of 1, add the ALN field by calling NSTORE during CON/REQ processing (after statement 7 and before calling NETPUT). See figure 3-13, CON/REQ Processing.

NETGETL

To get input on a list of ACNs use NETGETL; the calling format is

```
CALL NETGETL (aln,ha,ta,tlmax)
```

where:

aln	indicates one of the ALNs chosen by the application and previously made known to NAM in response to its CON/REQ supervisory message.
ha,ta,tlmax	indicates header and text area, and maximum text length, as covered under NETGET.

An ALN of 0 is a valid list number. It differs from ALNs not equal to 0 by the fact that it inputs data both from ACNs of 0 and ACNs not equal to 0. ALNs not equal to 0 input data only from ACNs not equal to 0. To NETGET on an ALN of 0, all asynchronous supervisory messages are delivered before data from any ACN not equal to 0.

When the response to a CON/REQ supervisory message contains an ALN of 1, replace the call to NETGET with a call to NETGETL. This replacement is necessary only when data is input. For example, the revised sample program statement 10 is replaced by

```
          .
          .
          .
10        CALL NETGETL (1,HA,TA,1).
          .
          .
          .
```

Changing the NETGET call at statement 6 of the revised sample program to a NETGETL is not necessary since it deals only with asynchronous supervisory messages.

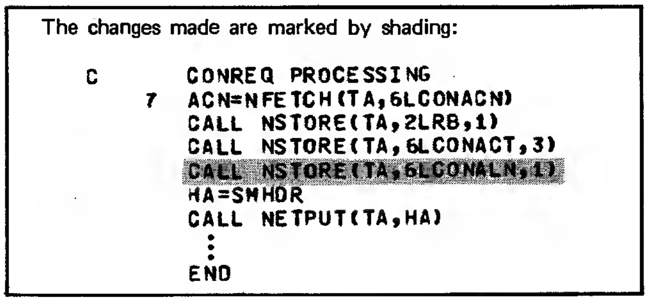
The changes made are marked by shading:

```
       PROGRAM EASY (OLTPUT)
      INTEGER PFCSFC,CONREQ,FCINIT,S
      INTEGER SMHDR,DTHDR,HA,TA(63)
C     SET UP CONSTANTS
      SMHDR=0
      CALL NSTORE(SMHDR,6LABHABT,3)
      CALL NSTORE(SMHDR,6LABHACT,1)
      CALL NSTORE(SMHDR,6LABHTLC,1)
      DTHDR=0
      CALL NSTORE(DTHDR,6LABHABT,2)
      CALL NSTORE(DTHDR,6LABHACT,4)
      CALL NSTORE(DTHDR,6LABHTLC,10)
      CONREQ=NFETCH(0,6LCONREQ)
      FCINIT=NFETCH(0,6LFCINIT)
      CALL NETON(5HMYAPP,NSUP,NSTAT,1,10)
      S=SHIFT(1,55)
      I=SHIFT(1,56)
      IF (NSTAT.EQ.0)GOTO 4
      PRINT 100,NSTAT
  100 FORMAT (*NETON FAILED,NSTAT=*,020)
      CALL NETOFF
      STOP 111
        ⋮
    5 CALL NETWAIT(4095,0)
C     CHECK FOR A SUP. MESSAGE OR INPUT AVAILABLE
      IF(NSUP.AND.S) 6,3
    3 IF(NSUP.AND.I) 10,5
C     SUP. MESSAGE IS AVAILABLE
    6 CALL NETGET(0,HA,TA,63)
      PFCSFC=NFETCH(TA,6LPFCSFC)
      IF (PFCSFC.EQ.CONREQ) GO TO 7
      IF (PFCSFC.EQ.FCINIT) GO TO 8
      PRINT 101,HA,TA
  101 FORMAT(*UNKNOWN SUP. MESSAGE *,64(/,1X,020/))
      CALL NETOFF
      STOP 222
C     CONREQ PROCESSING
    7 ACN=NFETCH(TA,6LCONACN)
      CALL NSTORE(TA,2LRB,1)
      CALL NSTORE(TA,6LCONACT,3)
      HA=SMHDR
      CALL NETPUT(HA,TA)
        ⋮
      GO TO 5
C     FC/INIT PROCESSING
    8 ACN=NFETCH(TA,5LFCACN)
      CALL NSTORE(TA,2LRB,1)
      HA=SMHDR
      CALL NETPUT(HA,TA)
        ⋮
      TA=10H INPUT PLS
      TA(2)=0
      HA=DTHDR
      CALL NSTORE(HA,6LABHACN,ACN)
      CALL NETPUT(HA,TA)
      GO TO 5
C     INPUT IS AVAILABLE
   10 CALL NETGET(ACN,HA,TA,1)
        ⋮
      END
```

Figure 3-11. Sample Program EASY Revised

	59	51	50	49	43	35	23	9	5 0
symbolic name	CON	EB	RB	REQ		CONACN		ACT	ALN
value	63_{16}	0	1	00_8					

Figure 3-12. Field of Application List Number (ALN)


The changes made are marked by shading:

```
C       CONREQ PROCESSING
      7 ACN=NFETCH(TA,6LCONACN)
        CALL NSTORE(TA,2LRB,1)
        CALL NSTORE(TA,6LCONACT,3)
        CALL NSTORE(TA,6LCONALN,1)
        HA=SMHDR
        CALL NETPUT(TA,HA)
          :
        END
```

Figure 3-13. CON/REQ Processing

In summary, the differences between NETGET and NETGETL are as follows:

NETGET

- Inputs data only on a specific ACN, which can be one of the following:

  A specific terminal

  A certain application

  An asynchronous supervisory message on ACN=0

- Returns the null block if there is no input available on the specified ACN.

NETGETL

- Inputs data in a round-robin fashion; each call inputs data from a different ACN. The ACNs from which data was input were previously grouped by specifying an ALN in response to every CON/REQ supervisory message from NAM.

- Accesses asynchronous supervisory messages only on an ALN of 0. All asynchronous supervisory messages are delivered before any data from ACNs not equal to 0.

- Skips ACNs that do not have input available. The null block is returned only if no ACN in the list has input available.

The first sample programs recognized only two supervisory messages, the CON/REQ and the FC/INIT. The application can also initiate more requests to NAM enabling it to have better control and overall greater flexibility. Other supervisory messages and extra features supported by NAM are explained in detail in the following sections. Appendix C provides a complete list of all supervisory messages and their meanings.

LOGICAL CONNECTION BREAKAGE

A logical connection can be broken at any time by either the application or by NAM. The application can either initiate disconnection or be informed of disconnection by NAM.

APPLICATION INITIATES THE DISCONNECT

If the application is to be disconnected from a specific logical connection, it sends the CONnection/END supervisory message, as shown in figure 4-1. The disconnected ACN is now free, and is available for reassignment.

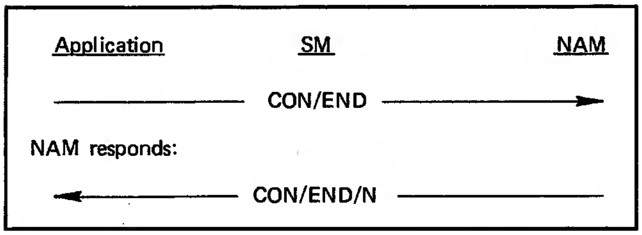
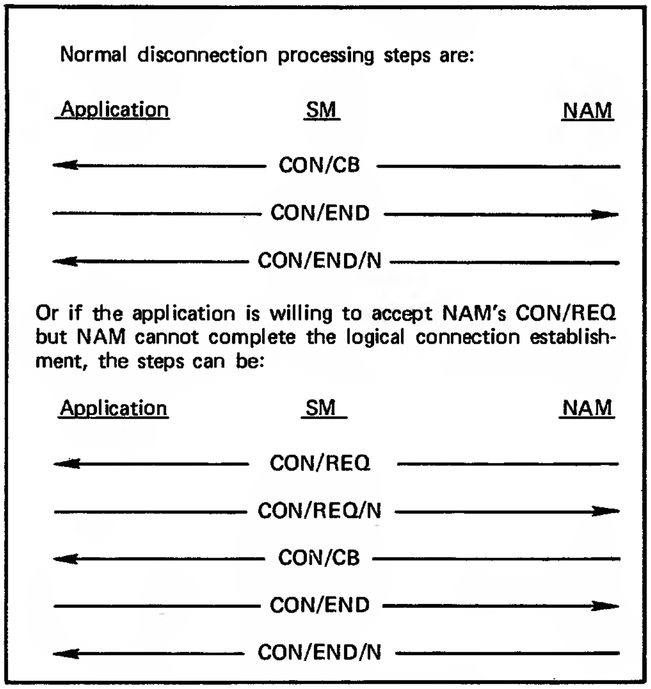


Figure 4-1. CONnection/END Supervisory Message

APPLICATION IS INFORMED OF THE DISCONNECT

In this case, the application receives a CON/CB SM from NAM, with a code giving the reason for the disconnect. After receipt of this message, the application should not NETPUT on that connection, but can still NETGET until a null block is received. The application then sends the CON/END SM. NAM responds by sending a CON/END/N response, after which the ACN is made free and can be reassigned to a new connection. See figure 4-2.


Normal disconnection processing steps are:

Or if the application is willing to accept NAM's CON/REQ but NAM cannot complete the logical connection establishment, the steps can be:

Figure 4-2. Connection Breakage Steps

The CON/CB and the CON/END formats are shown in figures 4-3 and 4-4, respectively. A new application can automatically be assigned after terminating the current application activities with the CON/END.

If application MYAPP is written to recognize a special type-in to terminate the connection (such as TERM), and allows the terminal user to type in the next application name (for example, TERM,IAF), application MYAPP can then be

59	51	50	49	43	35	23 0
CON	EB	RB	CB	RC	CONACN	
63_{16}	0	0	05_8	rc	acn	

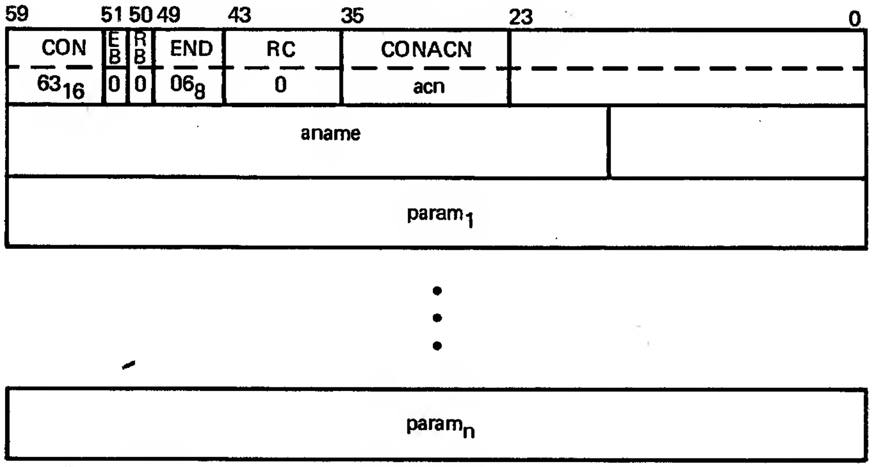
acn The ACN on which the connection was broken.

rc The reason code for breaking the connection and it can be:

1 = line disconnect

2 = connection broken by NAM (for example, a result of bad parameter on CON/REQ/N)

Figure 4-3. Format for Connection Broken (CON/CB)



acn The ACN to be ended.

aname 1 through 7 alphanumeric display code characters with a leading alphabetic character specifying the name of the application to which the connection should be switched.

$param_1$ to $param_n$ Parameters passed to the new application.

Figure 4-4. Format for Connection Ended (CON/END) Application to NAM

triggered to do a CON/END specifying IAF as the next application to which the terminal user is connected. Whether or not an application specifies the name of another application in its CON/END message, it receives a response message with the format shown in figure 4-5.

DOWNLINE FLOW CONTROL

Downline flow control involves acknowledgment/-nonacknowledgment of block delivery and stop versus break.

ACKNOWLEDGEMENT/NONACKNOWLEDGEMENT OF BLOCK DELIVERY

Because host computers are faster than terminals, it is possible for an application to send blocks along a particular connection faster than can be output at the terminal. It is also possible for the same application to send blocks so slowly that the terminal is underoccupied. Therefore, NAM provides a set of programming conventions allowing the application program to control the flow of data between the application and its terminals.

The application block limit (ABL) field is found in the CON/REQ supervisory message format. (See figure 3-4.) The ABL is established for each logical connection. It indicates how many blocks of data an application can have outstanding at the requested ACN at any given time.

As blocks are delivered to the terminal, an FC/ACK (acknowledge) supervisory message is returned to the application. If, for any reason, a block cannot be delivered, an FC/NAK (negative acknowledge) supervisory message is returned instead, and the block is lost. The application can attempt recovery by sending the lost block again. However, if more blocks have been sent following the lost block, the lost block might be delivered out of sequence.

The FC/ACK and the FC/NAK supervisory messages do not require responses from the application. The FC/ACK and FC/NAK formats are shown in figures 4-6 and 4-7, respectively.

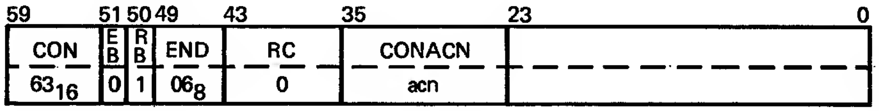


Figure 4-5. Format for Connection Ended (CON/END/N) NAM to Application

(NAM to Application)

59	51	50	49	43	35	23	5 0
FC	EB	RB	ACK		FCACN	FCABN	
83_{16}	0	0	02_8		acn	abn	

acn The ACN along which the block was delivered.

abn The ABN of the delivered block.

Figure 4-6. Format for Block Delivered (FC/ACK)

(NAM to Application)

59	51	50	49	43	35	23	5 0
FC	EB	RB	NAK	RC	FCACN	FCABN	
83_{16}	0	0	03_8	rc	acn	abn	

rc The reason code explaining why the block cannot be delivered:

1 = Lost block, (occurs only because of a system error)

acn The ACN along which a block cannot be delivered.

abn The ABN, which cannot be delivered.

Figure 4-7. Format for Block Not Delivered (FC/NAK)

If the number of blocks exceeds the value of ABL, an ERR/LGL (logical error) supervisory message is sent to the application, along with the reason for the error. The block to be delivered is discarded by NAM.

The following strategy is used to control the flow of blocks for delivery:

1. Define two vectors K(n) and M(n), where n is the largest ACN possible.

2. Set K(i)=0 and M(i)=ABL upon receipt of CON/REQ on ACN=i.

3. Set K(i)=K(i)-1 whenever an FC/ACK (block delivered) supervisory message is accepted for ACN=i.

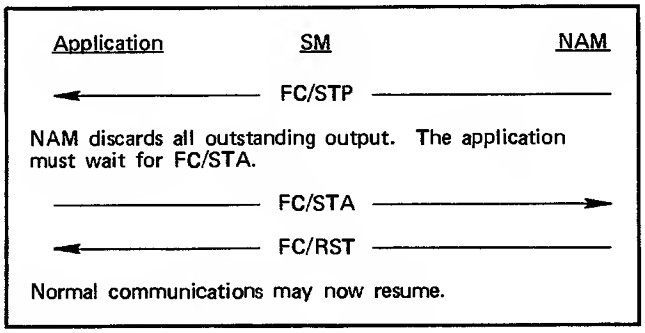
4. Set K(i)=0 whenever an FC/BRK (break) is received for ACN=i.

5. Set K(i)=K(i)+1 and output one block on ACN=i, when a block is to be output to NAM and K(i) < M(i). If K(i) ≥ M(i), wait until K(i) is decreased by step 3.

STOP VERSUS BREAK

Network conditions occasionally require the suspension of downline data traffic on a logical connection. When one of these conditions occurs, the application is notified to stop block delivery on the specified ACN until the condition has cleared.

If the NPU is able to determine when the condition has cleared, the NPU causes NAM to send an FC/STP (stop) supervisory message to the application, requesting it to suspend data traffic. The application then waits for an FC/STA (start) supervisory message from NAM. When the FC/STA is received from NAM, the application sends an FC/RST (reset) to enable traffic to resume. The FC/STP, FC/STA, and the FC/RST formats are shown in figures 4-8, 4-9, and 4-10, respectively.



NAM discards all outstanding output. The application must wait for FC/STA.

Normal communications may now resume.

Figure 4-8. FC/STP Protocol

If the NPU is not able to determine when the condition has cleared or, if a user break occurs, then an FC/BRK (break) supervisory message is sent to the application. When the FC/BRK is received from NAM, the application sends an FC/RST which enables traffic to resume. The FC/BRK format is shown in figure 4-11.

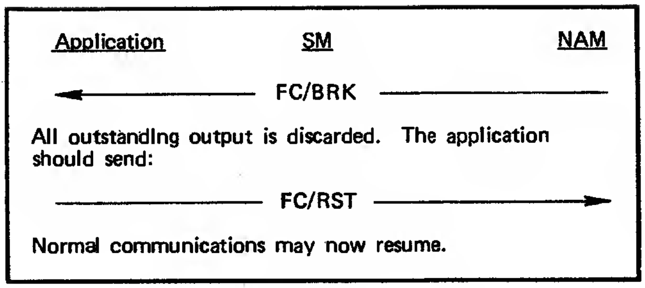

Application
SM
NAM
FC/BRK
All outstanding output is discarded. The application should send:
FC/RST
Normal communications may now resume.

Figure 4-9. FC/BRK Protocol

When a stop condition occurs, the following events take place:

1. The blocks that were output by the application but not acknowledged by NAM are discarded.

2. An FC/STP message with the following information is sent to the application:

   - The ACN of the stopped connection
   - The code giving the reason for the stop
   - The ABN of the last acknowledged downline block (zero if none)

3. The application uses NETGET to fetch any pending input until a null block on ACN or an FC/STA is received.

4. The application receives an FC/STA.

5. The application sends an FC/RST message to resume data traffic. The FC/RST message does not require a response from NAM.

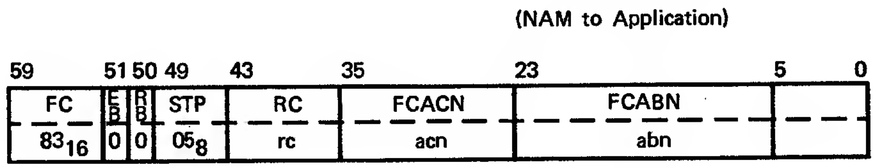
The FC/STP protocol is shown in figure 4-12.

When a break condition occurs, the following events take place:

1. The blocks that were output by the application but not acknowledged by NAM are discarded.

2. An FC/BRK message with the following information is sent to the application:

   - The ACN of the .connection on which the break condition occurreo
   - The code specifying the reason for the break
   - The ABN of the last acknowledged downline block (zero if none)



acn ACN of the stopped connection.

abn ABN of the last acknowledged block, zero if none.

rc Reason code for stop:

1 = Terminal busy

2 = Terminal failure

Figure 4-10. Format for Stop (FC/STP)

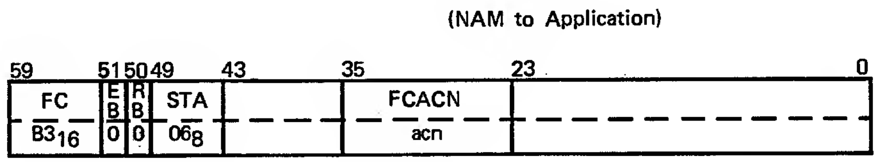


Figure 4-11. Format for Start (FC/STA)

3. The application sends an FC/RST to resume data traffic. The FC/RST message does not require a response from NAM.

4. The application uses NETGET to fetch any pending input until a null block is received.

The FC/BRK protocol is shown in figure 4-13.

INACTIVE CONNECTION AND HOST SHUTDOWN

Whenever NAM detects that 10 minutes have elapsed without any message traffic over a connection, it sends the FC/INA (flow-control/inactive) supervisory message to the application. Whether further action is performed on that connection depends on the coding of the application program. For example, the application can send a message to the terminal operator, disconnect the terminal, or do whatever is appropriate.

NAM continues sending the FC/INA supervisory message at 10-minute intervals, as long as there is no activity on the connection. The format of the FC/INA is shown in figure 4-14. The FC/INA message does not require a response from the application.

SHUT/INSD SUPERVISORY MESSAGES

The network operator can request idle down of network activities, or an immediate network shutdown by disabling the network. In either case, NAM sends the SHUT/INSD supervisory message to every application connected to it. The SHUT/INSD supervisory message is shown in figure 4-15.

If i is equal to 0, the application is allowed to end its current activities by issuing a CON/END message for every connection and a NETOFF. If i is equal to 1, immediate shutdown has been requested, and the application should issue NETOFF at once.

ERROR HANDLING

When the application sends an ill-formatted request to NAM, an ERR/LGL (logical error) supervisory message is queued for the application, informing it of the kind of error detected. The application inputs the ERR/LGLs queued for it, and takes action according to the error codes reported to it.

A counter is incremented each time an ERR/LGL is sent to the application. This procedure prevents deadlock situations when an application is outputting an infinite number of erroneous messages, and never obtaining the resulting ERR/LGLs queued for it. This counter is reset to zero every

(NAM to Application)

59	51	50	49	43	35	23	5 0
FC	EB	RB	BRK	RC	FCACN	FCABN	
83_{16}	0	0	00_8	rc	acn	abn	0

acn ACN of the connection on which a break condition occurred.

rc Reason code for break condition:

1 = User break 1. Caused by user input of the defined B1 key for his terminal, normally : .

2 = User break 2. Caused by user input of the defined B2 key for his terminal, normally) .

3 = Output device not ready.

4 = Incorrectly formatted data.

abn ABN of the last acknowledged block, zero if none.

Figure 4-12. Format for Break (FC/BRK)

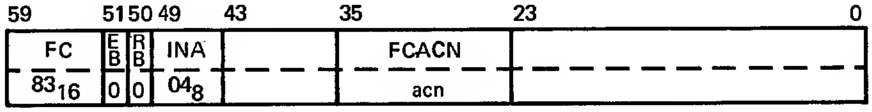
(Application to NAM)

59	51	50	49	43	35	23 0
FC	EB	RB	RST		FCACN	
83_{16}	0	0	01_8	0	acn	0

acn ACN of the connection to be reset.

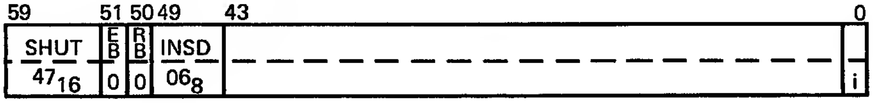
Figure 4-13. Format for Reset (FC/RST)

(NAM to Application)



acn ACN of the connection observed inactive.

Figure 4-14. Format for Flow-Control/Inactive (FC/INA)



i=0 The operator has requested an idle-down of network activities.

i=1 The operator is disabling the network.

Figure 4-15. Format for Host Shutdown (SHUT/INSD)

time the number of supervisory messages queued for the application is down to zero. Zero indicates that the application has read all of its outstanding supervisory messages. Once the counter has reached 100, NAM ignores any supervisory message resulting in another ERR/LGL until the counter of outstanding messages has decreased to zero again.

The counter mechanism protects NAM from accumulating an infinite number of ERR/LGLs queued for an application; however, it cannot make the application end its erratic behavior. An application can input all outstanding supervisory messages queued to it, including the ERR/LGLs, but disregard or misinterpret the ERR/LGLs and continue outputting an endless number of erroneous messages. In this case, the NAM counter never reaches 100 and the application remains in an erroneous loop.

The ERR/LGL format is shown in figure 4-16.

For greater detail about error codes, refer to the NAM 1 reference manual.

THE ECHO PROGRAM EXAMPLE

The ECHO program accepts every terminal requesting it, up to a maximum of 20. After connection is initialized, it sends the message INPUT PLS and then expects an input of any character string terminated by a carriage return (CR). It then echoes the string to the terminal, followed again by INPUT PLS, and waits for new input.

If one of the two user break keys defined for the terminal, normally the colon (:) or right parenthesis ( )), is hit, ECHO responds by sending BYE BYE and resumes normal echo mode. When more than 10 minutes have passed without terminal activity, the message TIMEOUT is sent to the terminal, which is then disconnected. An input of ten zeros from any terminal terminates ECHO.

The ECHO program is shown in figure 4-17.

(NAM to Application)

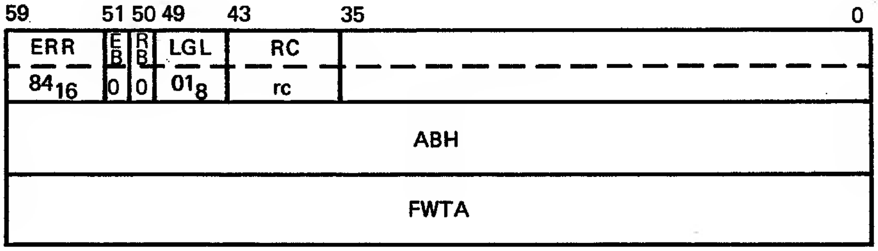


Figure 4-16. Format for Logical Error (ERR/LGL) (Sheet 1 of 2)

rc Reason code for error:

1 = Illegal ACT value

2 = Illegal TLC value

3 = Illegal ABT value

4 = Invalid ACN

5 = ABL exceeded

6 = NAM's counter of outstanding ERR/LGLs has reached 100. No more ERR/LGLs are sent, until the number of outstanding supervisory messages is zero again.

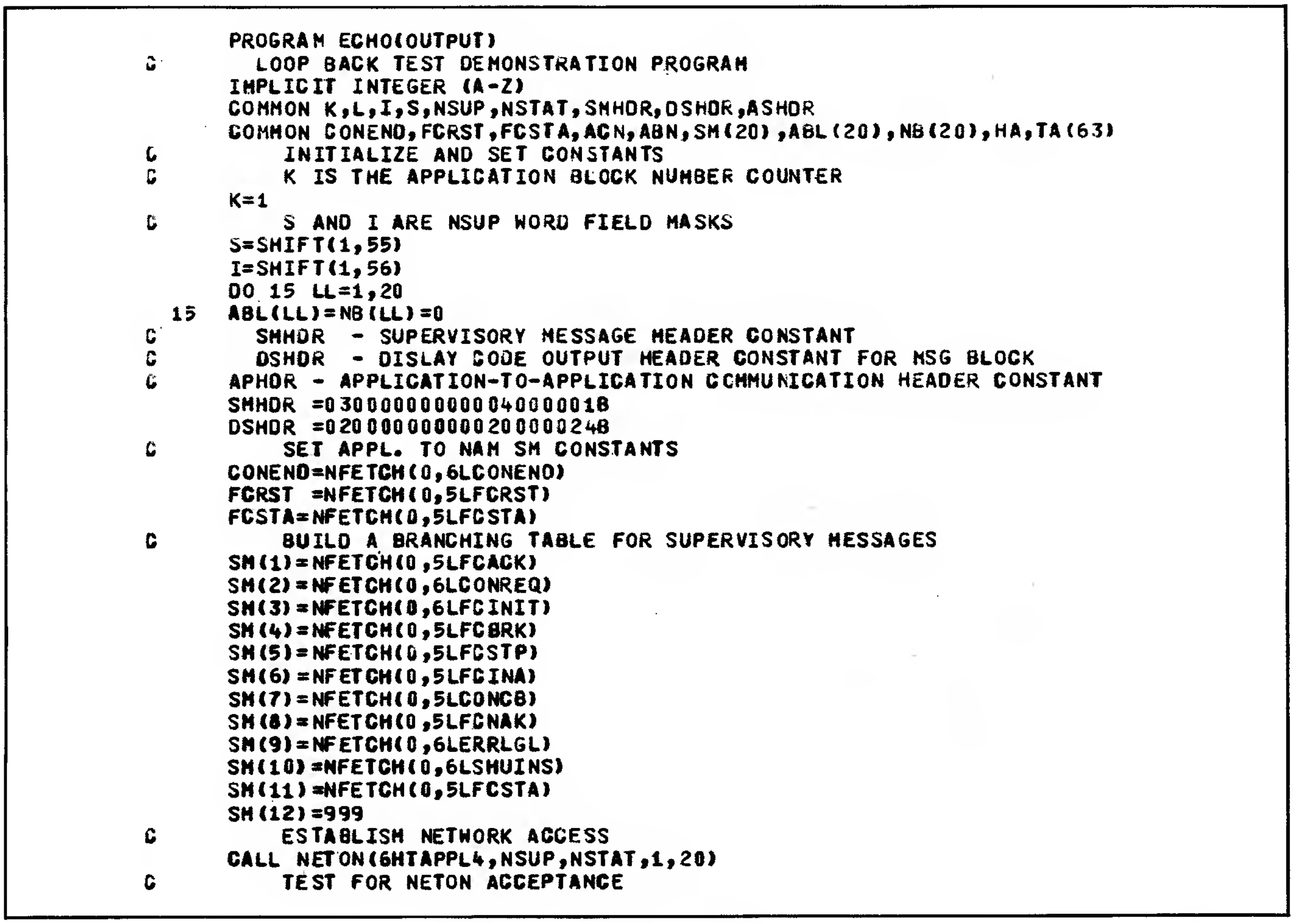
7 = Illegal or illogical supervisory message

8 = Fragmented input/output error

ABH If this word is nonzero, this is a copy of the ABH for the message causing the logical error.

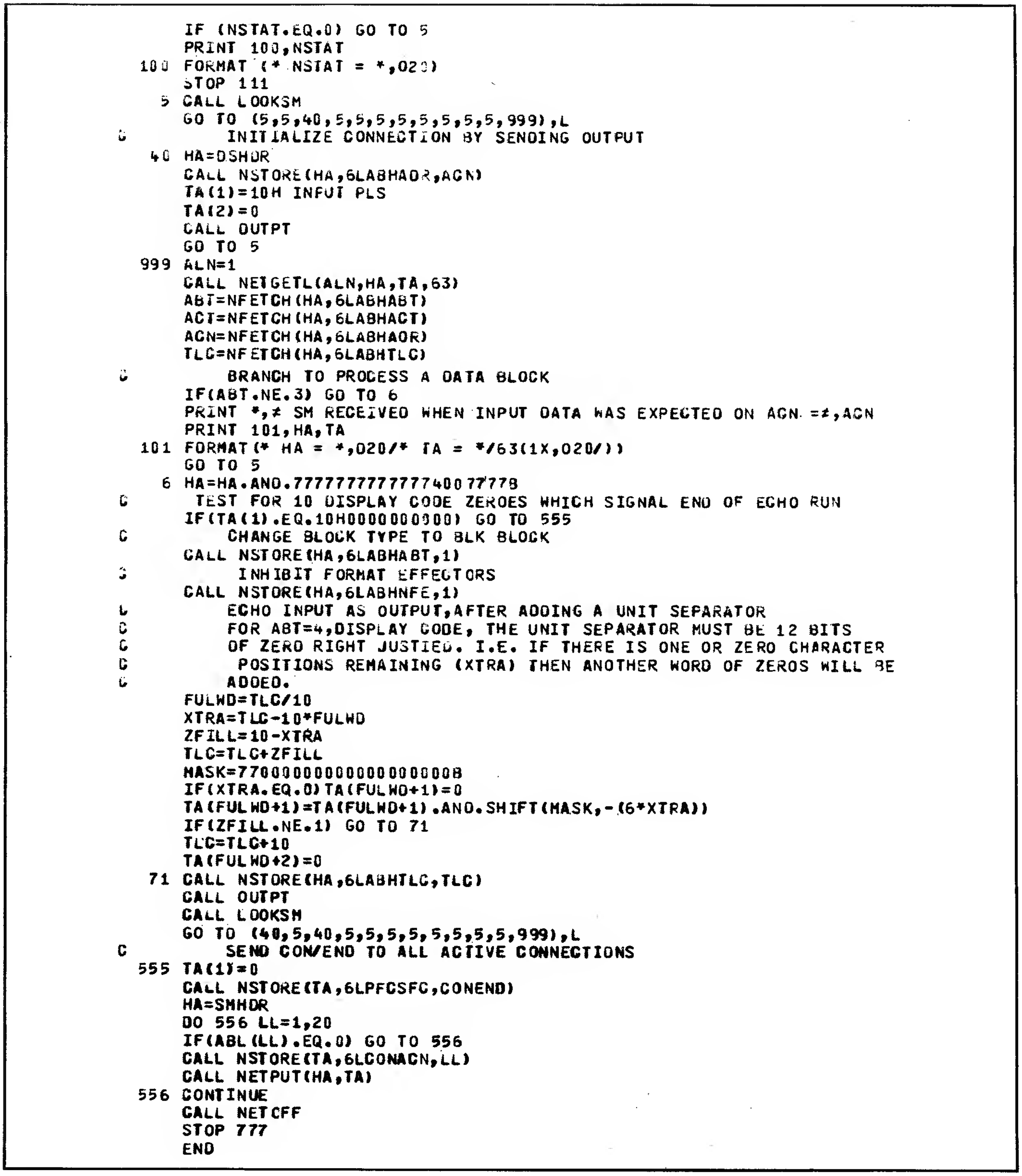
FWTA This is the first word of the text area of the message which caused the logical error. FWTA is sent only if ABH is nonzero.

Figure 4-16. Format for Logical Error (ERR/LGL) (Sheet 2 of 2)



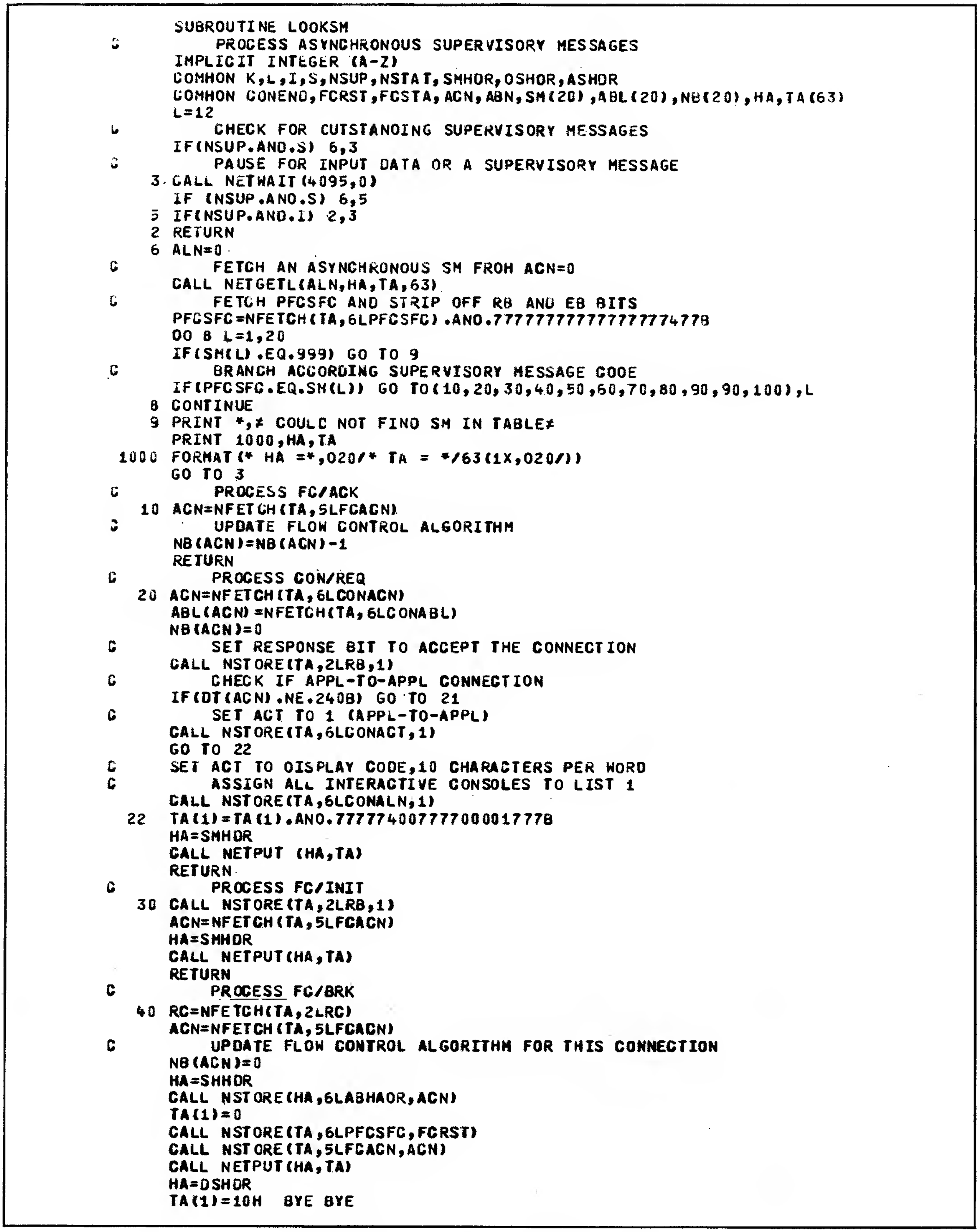
```
      PROGRAM ECHO(OUTPUT)
C       LOOP BACK TEST DEMONSTRATION PROGRAM
      IMPLICIT INTEGER (A-Z)
      COMMON K,L,I,S,NSUP,NSTAT,SMHDR,DSHDR,ASHDR
      COMMON CONEND,FCRST,FCSTA,ACN,ABN,SM(20),ABL(20),NB(20),HA,TA(63)
C         INITIALIZE AND SET CONSTANTS
C         K IS THE APPLICATION BLOCK NUMBER COUNTER
      K=1
C         S AND I ARE NSUP WORD FIELD MASKS
      S=SHIFT(1,55)
      I=SHIFT(1,56)
      DO 15 LL=1,20
   15 ABL(LL)=NB(LL)=0
C       SMHDR  - SUPERVISORY MESSAGE HEADER CONSTANT
C       DSHDR  - DISLAY CODE OUTPUT HEADER CONSTANT FOR MSG BLOCK
C     APHDR - APPLICATION-TO-APPLICATION CCMMUNICATION HEADER CONSTANT
      SMHDR =03000000000040000001B
      DSHDR =02000000000020000024B
C         SET APPL. TO NAM SM CONSTANTS
      CONEND=NFETCH(0,6LCONEND)
      FCRST =NFETCH(0,5LFCRST)
      FCSTA=NFETCH(0,5LFCSTA)
C         BUILD A BRANCHING TABLE FOR SUPERVISORY MESSAGES
      SM(1)=NFETCH(0,5LFCACK)
      SM(2)=NFETCH(0,6LCONREQ)
      SM(3)=NFETCH(0,6LFCINIT)
      SM(4)=NFETCH(0,5LFCBRK)
      SM(5)=NFETCH(0,5LFCSTP)
      SM(6)=NFETCH(0,5LFCINA)
      SM(7)=NFETCH(0,5LCONCB)
      SM(8)=NFETCH(0,5LFCNAK)
      SM(9)=NFETCH(0,6LERRLGL)
      SM(10)=NFETCH(0,6LSHUINS)
      SM(11)=NFETCH(0,5LFCSTA)
      SM(12)=999
C         ESTABLISH NETWORK ACCESS
      CALL NETON(6HTAPPL4,NSUP,NSTAT,1,20)
C         TEST FOR NETON ACCEPTANCE
```

Figure 4-17. The ECHO Program (Sheet 1 of 4)



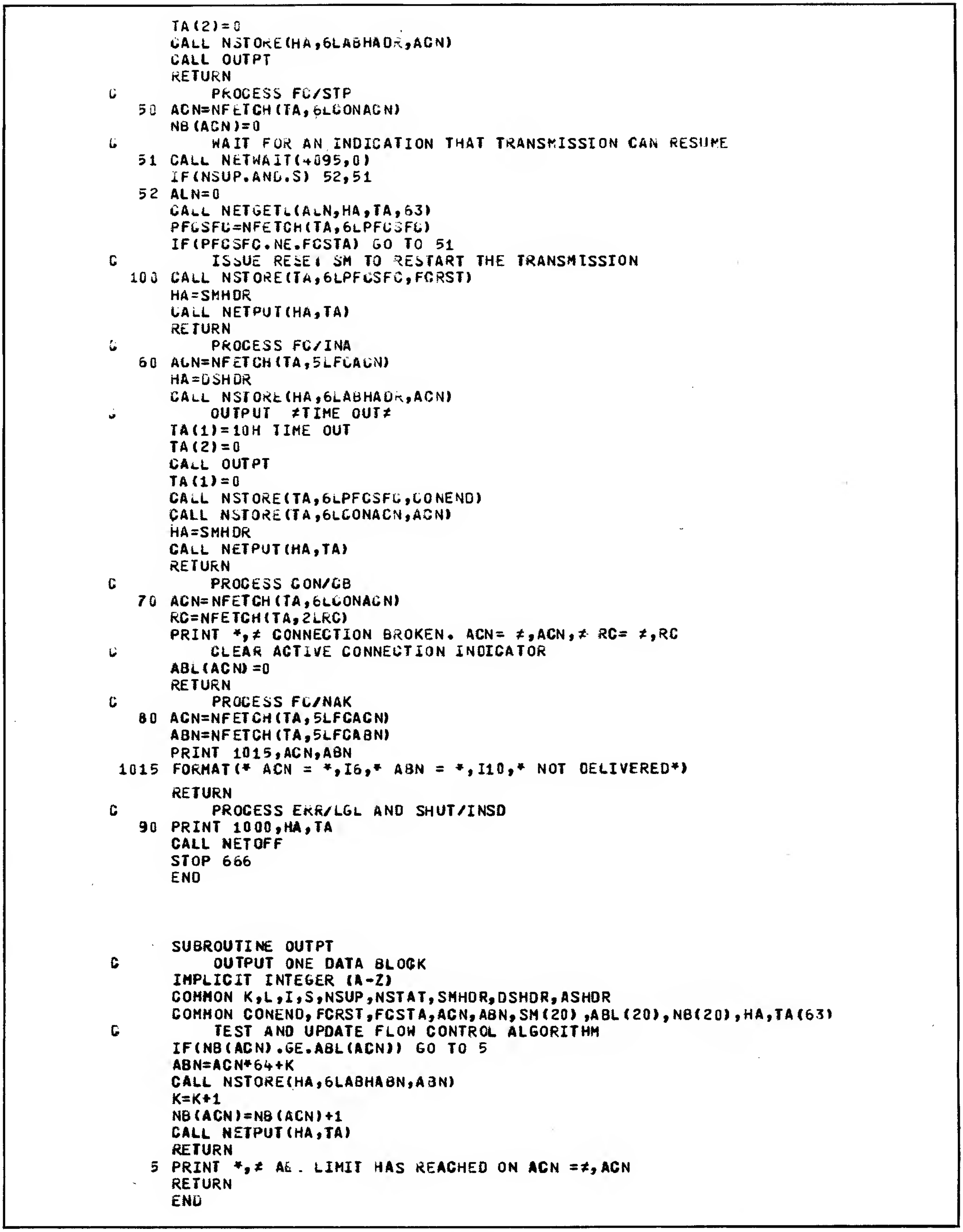
```
      IF (NSTAT.EQ.0) GO TO 5
      PRINT 100,NSTAT
  100 FORMAT (* NSTAT = *,O20)
      STOP 111
    5 CALL LOOKSM
      GO TO (5,5,40,5,5,5,5,5,5,5,5,999),L
C         INITIALIZE CONNECTION BY SENDING OUTPUT
   40 HA=DSHDR
      CALL NSTORE(HA,6LABHADR,ACN)
      TA(1)=10H INPUT PLS
      TA(2)=0
      CALL OUTPT
      GO TO 5
  999 ALN=1
      CALL NETGETL(ALN,HA,TA,63)
      ABT=NFETCH(HA,6LABHABT)
      ACT=NFETCH(HA,6LABHACT)
      ACN=NFETCH(HA,6LABHADR)
      TLC=NFETCH(HA,6LABHTLC)
C         BRANCH TO PROCESS A DATA BLOCK
      IF(ABT.NE.3) GO TO 6
      PRINT *,≠ SM RECEIVED WHEN INPUT DATA WAS EXPECTED ON ACN =≠,ACN
      PRINT 101,HA,TA
  101 FORMAT(* HA = *,O20/* TA = */63(1X,O20/))
      GO TO 5
    6 HA=HA.AND.77777777777774007777B
C       TEST FOR 10 DISPLAY CODE ZEROES WHICH SIGNAL END OF ECHO RUN
      IF(TA(1).EQ.10H0000000000) GO TO 555
C         CHANGE BLOCK TYPE TO BLK BLOCK
      CALL NSTORE(HA,6LABHABT,1)
C          INHIBIT FORMAT EFFECTORS
      CALL NSTORE(HA,6LABHNFE,1)
C         ECHO INPUT AS OUTPUT,AFTER ADDING A UNIT SEPARATOR
C         FOR ABT=4,DISPLAY CODE, THE UNIT SEPARATOR MUST BE 12 BITS
C         OF ZERO RIGHT JUSTIED. I.E. IF THERE IS ONE OR ZERO CHARACTER
C          POSITIONS REMAINING (XTRA) THEN ANOTHER WORD OF ZEROS WILL BE
C         ADDED.
      FULWD=TLC/10
      XTRA=TLC-10*FULWD
      ZFILL=10-XTRA
      TLC=TLC+ZFILL
      MASK=77000000000000000000B
      IF(XTRA.EQ.0)TA(FULWD+1)=0
      TA(FULWD+1)=TA(FULWD+1).AND.SHIFT(MASK,-(6*XTRA))
      IF(ZFILL.NE.1) GO TO 71
      TLC=TLC+10
      TA(FULWD+2)=0
   71 CALL NSTORE(HA,6LABHTLC,TLC)
      CALL OUTPT
      CALL LOOKSM
      GO TO (40,5,40,5,5,5,5,5,5,5,5,999),L
C         SEND CON/END TO ALL ACTIVE CONNECTIONS
  555 TA(1)=0
      CALL NSTORE(TA,6LPFCSFC,CONEND)
      HA=SMHDR
      DO 556 LL=1,20
      IF(ABL(LL).EQ.0) GO TO 556
      CALL NSTORE(TA,6LCONACN,LL)
      CALL NETPUT(HA,TA)
  556 CONTINUE
      CALL NETCFF
      STOP 777
      END
```

Figure 4-17. The ECHO Program (Sheet 2 of 4)



```
      SUBROUTINE LOOKSM
C         PROCESS ASYNCHRONOUS SUPERVISORY MESSAGES
      IMPLICIT INTEGER (A-Z)
      COMHON K,L,I,S,NSUP,NSTAT,SMHOR,OSHOR,ASHDR
      COMHON CONENO,FCRST,FCSTA,ACN,ABN,SM(20),ABL(20),NB(20),HA,TA(63)
      L=12
C         CHECK FOR CUTSTANOING SUPERVISORY MESSAGES
      IF(NSUP.ANO.S) 6,3
C         PAUSE FOR INPUT DATA OR A SUPERVISORY MESSAGE
    3 CALL NETWAIT(4095,0)
      IF (NSUP.ANO.S) 6,5
    5 IF(NSUP.AND.I) 2,3
    2 RETURN
    6 ALN=0
C         FETCH AN ASYNCHRONOUS SM FROH ACN=0
      CALL NETGETL(ALN,HA,TA,63)
C         FETCH PFCSFC AND STRIP OFF RB ANO EB BITS
      PFCSFC=NFETCH(TA,6LPFCSFC).ANO.77777777777777774778
      OO 8 L=1,20
      IF(SM(L).EQ.999) GO TO 9
C         BRANCH ACCORDING SUPERVISORY MESSAGE COOE
      IF(PFCSFC.EQ.SM(L)) GO TO(10,20,30,40,50,60,70,80,90,90,100),L
    8 CONTINUE
    9 PRINT *,≠ COULC NOT FINO SM IN TABLE≠
      PRINT 1000,HA,TA
 1000 FORMAT(* HA =*,020/* TA = */63(1X,020/))
      GO TO 3
C         PROCESS FC/ACK
   10 ACN=NFETCH(TA,5LFCACN)
C         UPDATE FLOW CONTROL ALGORITHM
      NB(ACN)=NB(ACN)-1
      RETURN
C         PROCESS CON/REQ
   20 ACN=NFETCH(TA,6LCONACN)
      ABL(ACN)=NFETCH(TA,6LCONABL)
      NB(ACN)=0
C         SET RESPONSE BIT TO ACCEPT THE CONNECTION
      CALL NSTORE(TA,2LRB,1)
C         CHECK IF APPL-TO-APPL CONNECTION
      IF(DT(ACN).NE.240B) GO TO 21
C         SET ACT TO 1 (APPL-TO-APPL)
      CALL NSTORE(TA,6LCONACT,1)
      GO TO 22
C     SET ACT TO OISPLAY CODE,10 CHARACTERS PER WORD
C         ASSIGN ALL INTERACTIVE CONSOLES TO LIST 1
      CALL NSTORE(TA,6LCONALN,1)
   22 TA(1)=TA(1).ANO.77777400777700001777B
      HA=SMHDR
      CALL NETPUT (HA,TA)
      RETURN
C         PROCESS FC/INIT
   30 CALL NSTORE(TA,2LRB,1)
      ACN=NFETCH(TA,5LFCACN)
      HA=SMHDR
      CALL NETPUT(HA,TA)
      RETURN
C         PROCESS FC/BRK
   40 RC=NFETCH(TA,2LRC)
      ACN=NFETCH(TA,5LFCACN)
C         UPDATE FLOW CONTROL ALGORITHM FOR THIS CONNECTION
      NB(ACN)=0
      HA=SHHDR
      CALL NSTORE(HA,6LABHAOR,ACN)
      TA(1)=0
      CALL NSTORE(TA,6LPFCSFC,FCRST)
      CALL NSTORE(TA,5LFCACN,ACN)
      CALL NETPUT(HA,TA)
      HA=DSHDR
      TA(1)=10H  BYE BYE
```

Figure 4-17. The ECHO Program (Sheet 3 of 4)



```
      TA(2)=0
      CALL NSTORE(HA,6LABHADR,ACN)
      CALL OUTPT
      RETURN
C         PROCESS FC/STP
   50 ACN=NFETCH(TA,6LCONACN)
      NB(ACN)=0
C         WAIT FOR AN INDICATION THAT TRANSMISSION CAN RESUME
   51 CALL NETWAIT(4095,0)
      IF(NSUP.AND.S) 52,51
   52 ALN=0
      CALL NETGETL(ALN,HA,TA,63)
      PFCSFC=NFETCH(TA,6LPFCSFC)
      IF(PFCSFC.NE.FCSTA) GO TO 51
C         ISSUE RESET SM TO RESTART THE TRANSMISSION
  100 CALL NSTORE(TA,6LPFCSFC,FCRST)
      HA=SMHDR
      CALL NETPUT(HA,TA)
      RETURN
C         PROCESS FC/INA
   60 ACN=NFETCH(TA,5LFCACN)
      HA=DSHDR
      CALL NSTORE(HA,6LABHADR,ACN)
C         OUTPUT  ≠TIME OUT≠
      TA(1)=10H TIME OUT
      TA(2)=0
      CALL OUTPT
      TA(1)=0
      CALL NSTORE(TA,6LPFCSFC,CONEND)
      CALL NSTORE(TA,6LCONACN,ACN)
      HA=SMHDR
      CALL NETPUT(HA,TA)
      RETURN
C         PROCESS CON/CB
   70 ACN=NFETCH(TA,6LCONACN)
      RC=NFETCH(TA,2LRC)
      PRINT *,≠ CONNECTION BROKEN. ACN= ≠,ACN,≠ RC= ≠,RC
C         CLEAR ACTIVE CONNECTION INDICATOR
      ABL(ACN)=0
      RETURN
C         PROCESS FC/NAK
   80 ACN=NFETCH(TA,5LFCACN)
      ABN=NFETCH(TA,5LFCABN)
      PRINT 1015,ACN,ABN
 1015 FORMAT(* ACN = *,I6,* ABN = *,I10,* NOT DELIVERED*)
      RETURN
C         PROCESS ERR/LGL AND SHUT/INSD
   90 PRINT 1000,HA,TA
      CALL NETOFF
      STOP 666
      END



      SUBROUTINE OUTPT
C         OUTPUT ONE DATA BLOCK
      IMPLICIT INTEGER (A-Z)
      COMMON K,L,I,S,NSUP,NSTAT,SMHDR,DSHDR,ASHDR
      COMMON CONEND,FCRST,FCSTA,ACN,ABN,SM(20),ABL(20),NB(20),HA,TA(63)
C         TEST AND UPDATE FLOW CONTROL ALGORITHM
      IF(NB(ACN).GE.ABL(ACN)) GO TO 5
      ABN=ACN*64+K
      CALL NSTORE(HA,6LABHABN,ABN)
      K=K+1
      NB(ACN)=NB(ACN)+1
      CALL NETPUT(HA,TA)
      RETURN
    5 PRINT *,≠ AL. LIMIT HAS REACHED ON ACN =≠,ACN
      RETURN
      END
```

Figure 4-17. The ECHO Program (Sheet 4 of 4)

DATA FORMATS AND TERMINAL CONTROL 5

THE INTERACTIVE VIRTUAL TERMINAL

An interactive virtual terminal (IVT) is defined as a logical device that sends or receives logical lines of ASCII characters. These logical lines are transformed by the IVT interface software to physical lines of characters of the appropriate code set for the real terminal. IVT eliminates the application program of having to be concerned with which real terminals it is connected to and what specific characteristics they have (for example, line length, page length, control capabilities). IVT relieves the application from having to provide separate procedures in order to functionally service a variety of real terminals.

The choice of functions provided by an IVT is deliberately restricted to ensure efficient implementation, even when transferring data to a real terminal with the lowest capabilities. The following IVT characteristics can be assumed by the application:

- Infinite line width
- Infinite page size
- Use of the ASCII 128-character set

When an application requires features that are not provided by the IVT, but known to exist on the connected real terminal, the application can perform the following functions:

- Embed appropriate control characters in the output text, or scan for significant control characters in the input text.
- Transfer data in transparent mode (in which case, the application has direct access to, and responsibility for, all real terminal features).

IVT MODE TRANSFORMS AND FORMAT EFFECTORS

As already explained, when working in IVT mode, logical lines are transformed by the IVT interface to physical lines of the appropriate code set, according to the real terminal characteristics. These are known as IVT mode transforms.

These transforms are explained in greater detail in the following subsections.

INPUT AND OUTPUT OPERATION

ASCII is the recommended character type, although display code can be used. All 128 ASCII characters are valid, and the 7-bit ASCII characters are right-justified in either 8-bit or 12-bit bytes, as specified by the ACT field of the ABH. (See section 2.) The parity bit in both the 8-bit and 12-bit bytes is always set to zero by IVT transforms of the network. On output, NAM strips the upper 4 bits of the 12-bit bytes and sets them to zero on input. When using display code characters, only the characters of the installation character set are valid. They occupy 6-bit frames, 10 characters per word. Note that the use of display code characters involves a character-by-character conversion process, which significantly increases processing overhead.

INPUT

The line feed (LF) key signifies the end of a physical line, and when it is detected (or when the number of data characters reaches the page width defined for the real terminal), the preceding data is sent to the application as a BLK type block. (See section 2.)

The carriage return (CR) key signifies the end of a logical line. When it is detected, the preceding data is sent to the application as a MSG type block. Either delimiter is discarded.

A logical line compr ses a message; it can consist of one or more BLK blocks for every physical line, followed by a MSG block for the last line terminated by a CR.

OUTPUT

One or more logical lines can comprise a BLK or a MSG type block, subject to the restriction that a logical line cannot span block boundaries.

The application terminates logical lines by one of the following:

- A unit separator character (IF$_{16}$) when using ASCII characters
- 12 to 66 bits of binary zeros, right-justified in one or two 60-bit words, when using display code characters

The user can also control output lines by using format effectors.

FORMAT EFFECTORS

Format effectors are data characters that prefix each logical output line, but are not printed or displayed. They provide a means of achieving device-independent format control.

Format effectors are the same for either ASCII or display code output. They are interpreted as follows:

blank	Space 1 line before printing (note 1).
0	Space 2 lines before printing (note 1).
-	Space 3 lines before printing (note 1).
+	Position to start of the current line before printing.
*	Position to top of form or home cursor before printing.
1	Position to top of form or home cursor, and clear the screen before printing.
,	Perform no action.

.   Space 1 line after printing.

/   Position to start of the current line after printing.

NOTE

1. If the previous operation was input, the number of spaces is reduced by 1.

2. Any characters other than the preceding format effectors are treated as blanks.

Format effectors do not exist if the no format effector (NFE) bit of the ABH word is set to 1. See section 5.

LF and CR characters can be embedded anywhere within an ASCII output logical line. They are interpreted as follows:

LF   Cursor down one line.

CR   Return cursor to start of the current line.

THE TRANSPARENT MODE

Transparent mode can be seen as the opposite of IVT mode. In transparent mode, IVT transforms are inhibited; for example, all 256 possible 8-bit characters can be used without conversion or interpretation by the network software.

While in transparent mode, the application has complete freedom and responsibility with regard to data formatting and terminal control; IVT format effectors, code conversion, and paging are not performed. Auto-input still applies, and page-waiting occurs on message boundaries. Transparent mode is selected differently for output and input operations.

On output, it is selected by setting the transparent mode/IVT mode indicator (XPT) bit in the ABH word. The transparent mode is then in effect until TLC characters are transferred (see section 5). After that, normal IVT mode resumes.

On input, it is selected either by the application program or by the terminal user. In both cases, transparent input mode is terminated by one of the three possible methods:

- By a specific character code identification
- By a character limit count
- By a character timeout of 200 to 400 milliseconds

The CTRL/DEF SSM is used to select and delimit transparent input data. (For further information, refer to section 5.)

The parity bit, as selected by the parity (PA) option, determines the actual character codes received by the application. For a PA of N, all 256 8-bit character codes are received; for all other parity settings, the eighth bit is set to zero by the network.

CHARACTER TYPE CONTROL

When an application program changes its output character type, it changes the ACT field in the ABH to the appropriate character type. (See section 2.) When an application program changes its input character type, it sends the data control/change input character type (DC/CICT) supervisory message to NAM. (See appendix C.) The initial input character type is specified in the ACT field of the normal response to the CON/REQ SM. The format for the DC/CICT SM is shown in figure 5-1. The DC/CICT does not require a response from NAM.

TERMINAL CHARACTERISTICS

An application can redefine or modify certain terminal characteristics and parameters, where those of the connected real terminal differ from the assumptions made for the IVT mode. The application sends the CTRL/DEF (terminal characteristic redefinition) SM to NAM. It has the format shown in figure 5-2.

SPECIFYING INDIVIDUAL CHARACTERISTICS

The following character strings are used as terminal definition declarations.

PW=nnn   Specifies the physical page width of the terminal to be nnn decimal characters.

On output, the network software advances the cursor to the beginning of the next physical line, where the number of characters transmitted equals the specified page width.

The nnn parameter must be in the range of 0 to 255, where nnn=0 means never advance to a new line as a result of a page width.

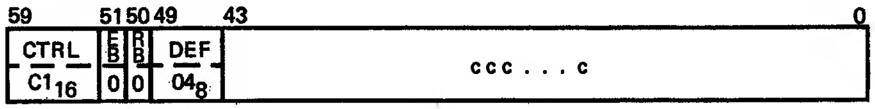
(Application to NAM)

59	51	50	49	43	35	23	5 0
DC	EB	RB	CICT		DCACN		ACT
$C2_{16}$	0	0	00_8	0	acn	0	act

acn   The ACN from which a new character type input is requested.

act   The new character type code (as defined in section 2).

Figure 5-1. Format for Change Input Character Type (DC/CICT)



ccc . . . c A string of ASCII characters in the form:

kk=param Also the form used for specifying terminal control commands.

kk A two-character keyword called the terminal descriptor.

param A parameter associated with the keyword to define a certain terminal characteristic.

Figure 5-2. Format for Terminal Characteristic Redefinition (CTRL/DEF)

PL=nnn Specifies the number of physical lines on a page. On output, the network software advances the cursor to the next page when the number of physical lines transmitted equals the page length.

The nnn parameter must be in the range of 0 to 255, where 0 means no paging is desired.

PA=t Specifies which type of parity to expect on input, and which to generate on output. The t parameter can have the following values:

Z Sets parity bit to zero.

O Sets or checks for odd parity.

E Sets or checks for even parity.

N Is an information bit, no parity.

The difference between PA equal to Z and PA equal to O or E is that for PA equal to Z, no parity check calculation is made and the bit must always be zero; for PA equal to O or E a parity check is made by the Network Processing Unit (NPU), but to the application the parity bit is always passed as a zero. The reason for this is to form the same ASCII character bit-pattern in memory, whether it comes from an even or odd parity terminal.

On input operation, the eighth bit is not set to 0 when PA is equal to N and while the operation is in transparent mode. In all other parity settings, the eighth bit is set to zero, regardless of the parity checking performed.

BS=c Specifies c as the backspace character associated with the terminal. Whenever c is detected, the last character entered is deleted from the input buffer. The deleted character and the selected backspace character are not transmitted to the application.

B1=x Specifies x as the character for user break-1. It means that when x (CR) is sent by the user, it causes NAM to send to the application the FC/BRK SM, specifying that a user break-1 has occurred. (See section 4.)

B2=x Specifies x as the character for user break-2. It means that when x (CR) is sent by the user, it causes NAM to send to the application the FC/BRK SM, specifying that a user break-2 has occurred.

CN=x Specifies x as the character to be used to cancel the input logical line in progress. It means that whenever x is entered as the last character of an input line (followed by a (CR)), and any portion of the logical line has already been sent to the application, a MSG block is sent with the cancel bit set. The application is notified to cancel the last entered input line. See section 5.

PG=t The t parameter specifies whether or not page-waiting is in effect. It can have the following values:

Y Page-waiting is in effect.

N Page-waiting is not in effect.

When output is sent to a CRT and page-waiting is in effect, the Terminal Interface Program (TIP) pauses its output display at each page boundary, and waits for any input followed by a (CR) to be typed in, before it displays the next output page.

If the page is not full and more output is available, OVER is displayed on the next available line. The next available output is displayed only after input followed by a (CR) is entered by the user.

NOTE

A null input line can be entered by the user; for example, when a page of output is to be acknowledged. In order to do so, the user must enter ,

x (CR)

or simply (CR)

where x is the control character defined for the terminal.

If page-waiting is in effect when the null line is entered, the line signals output acknowledgment only, and the line is not sent to the application. If, however, page-waiting is not in effect, the line is sent to the application as a MSG block.

Only one terminal control command can be specified for each CTRL/DEF SSM. If the PA and BS commands are needed, they are requested in separate supervisory messages. For example:

```
59      51 50 49     43                      0
| CTRL  | 0 | 0 | DEF |         PA = E        |
```

followed by

```
59      51 50 49     43                      0
| CTRL  | 0 | 0 | DEF |         BS = *        |
```

SELECTING TERMINAL CLASS

In order to associate a terminal with a set of characteristics, a terminal class is specified. A terminal class is a number, 1 through 15, which identifies a specific physical terminal type, and associates it with a set of default terminal characteristics. There are 14 terminal classes and they are shown in table 5-1.

TABLE 5-1. TERMINAL CLASSES

Terminal Type	Terminal Class
M33	1
713	2
2741	4
M40	5
H2000	6
751	7
T-4014	8
HASP	9
200UT	10
214	11
711	12
714	13
731	14
734	15

Every class has its associated set of default terminal characteristics. For example, a few of the associated characteristics of the 713 terminal are as follows:

PW (page width) = 72

PL (physical number of lines) = 0, indicating an infinite page length

PA (parity) = E

BS (backspace character) =←

B1 (user break-1 character) = :

B2 (user break-2 character) =)

CN (cancel input line character) = $

For a complete list of terminal characteristics, refer to the NAM 1 reference manual.

To select terminal class (TC), use the following command:

TC=n Specifies the terminal as belonging to terminal class n.

For example, to select the 214 console, send

```
59      51 50 49     43                      0
| CTRL  | 0 | 0 | DEF |        TC = 11        |
```

The device type (DT) is associated with the terminal class and is used to identify batch devices. The device type is an identifier passed to the application by the CON/REQ SM. (Refer to figure 3-4.) It is also used to identify application-to-application connections. Refer to section 6.

The device type (DT) codes are as follows:

0	Console
1	Card reader
2	Line printer
3	Card punch
4	Plotter
5	Application-to-application

NOTE

The combined field of DT and TC in the CON/REQ SM is also called the device type, and is accessed by the CONDT keyword. (Refer to figure 3-4 and appendix B.)

SELECTING AND DELIMITING TRANSPARENT MODE INPUT

The IN= descriptor is used for selecting the input device in either an IVT or transparent mode. The IN= parameters are as follows:

KB Input device is a keyboard in IVT mode.

PT Input device is a paper tape in IVT mode.

XK Input device is a keyboard in transparent mode.

XP Input device is a paper tape in transparent mode.

If the transparent mode is selected, the next input operation is in transparent mode, until a transparent mode delimiter occurs.

The delimiter is specified by

DL=Xhh,Cnnnn,TO

where:

hh indicates two hexadecimal digits representing a delimiter that terminates transparent mode input.

nnnn indicates 1 through 4096 maximum character count for transparent mode.

TO indicates input character timeout of 200 to 400 milliseconds.

NOTE

At least one delimiter must be specified. If more than one is selected, transparent input mode is terminated by the delimiter that occurs first.

TERMINAL CHARACTERISTICS CHANGED BY THE USER

A terminal user can define or modify certain terminal characteristics.

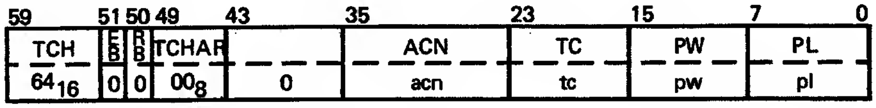
Whenever terminal class (TC), page width (PW), or page length (PL) are redefined by the user, the application is notified of the change by NAM. NAM sends to it the TCH/TCHAR SM (terminal characteristics redefined) shown in figure 5-3.

NOTE

- Other redefinitions are not communicated to the application.
- No response to TCH/TCHAR is required.

ABH FLAGS

There is one set of flags applicable for input operation, and another set applicable for output operation. The flags for input are shown in figure 5-4; the flags for output are shown in figure 5-5.



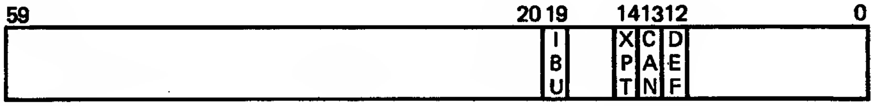
acn ACN for which the user has redefined TC, PW or PL:

tc Terminal class

pw Page width

pl Page length

Figure 5-3. Format for Terminal Characteristics Redefined (TCH/TCHAR)



PEF Parity error flag:

PEF=1 A parity error was detected in one or more input characters of the block transmitted.

PEF=0 No parity error was detected.

CAN Cancel flag:

CAN=1 Indicates that the message being received on this connection has been cancelled by the terminal user; for example, one or more BLK blocks of the message were previously transmitted to the application, and have been discarded.

CAN=0 Otherwise.

Figure 5-4. Input ABH Flag Format (Sheet 1 of 2)

XPT Transparent mode/IVT mode indicator:

XPT=1 Transparent mode, IVT transforms are inhibited. This mode is applicable to ACT 2 or 3 only. When an attempt is made to input in transparent mode and the ACT is not 2 or 3, the IBU bit sets, and the block is not transmitted. (See also IBU bit.)

XPT=0 Normal IVT mode, transforms are in effect.

IBU Input block undeliverable:

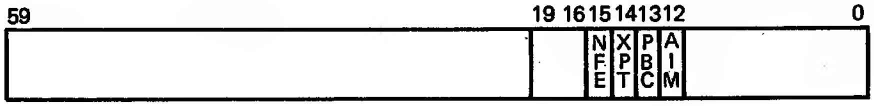
IBU=1 An attempt was made to input a block which is larger than the maximum text area. (Refer to section 3). In such a case, the block is not transmitted, but rather remains queued within NAM until another AIP input call is performed, specifying enough length to accommodate the block.

The true length of the block is found in the TLC field of the ABH.

As mentioned in explaining the XPT bit, it is also set to 1 if transparent mode input is requested and the ACT is not 2 or 3.

IBU=0 The block was transmitted to the text area.

Figure 5-4. Input ABH Flag Format (Sheet 2 of 2)



AIM Auto input mode indicator:

AIM=1 Auto input mode is requested. It means that this data block is to be sent to a terminal as output, and its first 20 characters are to form the subsequent input from that terminal. The output block must be an ABT of 2 (other ABTs are ignored). The block can be output in transparent mode, but the corresponding input block is not transparent, unless explicitly declared.

AIM=0 Otherwise, normal mode.

PBC Punch banner card:

PBC=0 Otherwise, no punched banner card is requested.

XPT Transparent mode/IVT mode indicator:

XPT=1 Specifies the block to be output is in transparent mode, in which case IVT transforms are inhibited. Normal IVT mode resumes after TLC characters have been transferred. The XPT=1 is ignored if ACT is not 2 or 3.

XPT=0 Normal IVT mode, transforms are in effect.

NFE No format effectors. Applies to data in an IVT mode only.

NFE=1 Specifies that no format effectors are to be associated with the data block. The output line is single-spaced before printing, and the first character is printed or displayed.

NFE=0 Format effectors are present in the output data block (for example, the line is spaced according to the first character, which is not printed or displayed).

Figure 5-5. Output ABH Flag Format

MISCELLANEOUS FEATURES 6

In this section, the following miscellaneous features of NAM are presented: the parallel mode, fragmented buffer routines, application-to-application communication, the debug option, and the statistical option.

THE PARALLEL MODE

In its normal mode of operation, NAM returns control to the calling application only after a full completion of the requested operation has occurred. In the parallel mode, however, the application can continue execution before the last NAM request has been completed.

To check the status of the last request, NAM sets the flag bits in the nsup word. NAM checks the c bit in the nsup word for parallel mode processing.

To select the parallel mode of operation, call the NETSETP routine:

CALL NETSETP (0)

To terminate the parallel mode of operation, call the NETSETP routine:

CALL NETSETP (1) or any other value not equal to 0

In parallel mode, NETCHEK must be called before checking the c bit:

CALL NETCHEK

NETCHEK performs two functions:

It updates the c bit in the nsup word. See figure 6-1.

It attempts to reissue the previous call to AIP if it was rejected by NAM.

If the c bit is not set, NETCHEK must be called to reflect the status of the last request. While the c bit is not set, only NETCHEK or NETWAIT calls are allowed.

NETCHEK must be called and the c bit checked for a completion status before any of the following routines or NETWAIT can be called in again:

NETGET

NETGETL

NETGETF

NETGTFL

NETPUT

NETPUTF

When NETWAIT is called in parallel mode, the program is suspended and the parallel mode of operation is ignored. The parallel mode is reinstated when the NETWAIT call is completed.

Sample program EASY, as shown in figure 6-2, provides an example of parallel mode operation.

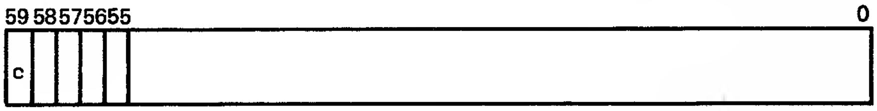
THE FRAGMENTED BUFFER ROUTINES

The fragmented buffer routines perform data transfer from a buffer that is split into fragments.

Data for input is split into independent fragments. Data for output is gathered from the fragments to constitute one complete block. The fragmented buffer routines are NETGETF and NETGTFL for input, and NETPUTF for output. They are in all respects similar to NETGET, NETGETL and NETPUT, respectively; only their text area is split into fragments.

There can be up to 40 fragments each having its own length as chosen by the application.

The independent fragments make better use of memory space. They gather memory space that is created when messages are processed in a different order than they have arrived. Although every fragment has its own length, situations of a variable size buffer management processing are rare. Therefore, the fragmented buffer mechanism is recommended for applications using a pool of fixed length buffer areas.



The c bit is applicable only when working in the parallel mode, and its meaning is:

c Complete bit:

1 = Last NAM call has been completed

0 = Otherwise

Figure 6-1. NSUP Word Parallel Mode Bit

The changes made are marked by shading:

```
         PROGRAM EASY(OUTPUT)
         INTEGER PFCSFC,CONREQ,FCINIT,S,C
         INTEGER SMHDR,DTHDR,HA,TA(63)
C        SET UP CONSTANTS
         SMHDR=0
         CALL NSTORE(SMHDR,6LABHABT,3)
         CALL NSTORE(DTHDR,6LABHACT,1)
         CALL NSTORE(SMHDR,6LABHTLC,1)
         DTHDR=0
         CALL NSTORE(DTHDR,6LABHABT,2)
         CALL NSTORE(DTHDR,6LABHACT,4)
         CALL NSTORE(DTHDR,6LABHTLC,10)
         CONREQ=NFETCH(0,6LCONREQ)
         FCINIT=NFETCH(0,6LFCINIT)
         S=SHIFT(1,55)
         I=SHIFT(1,56)
         C=SHIFT(1,59)
         CALL NETON(5HMYAPP,NSUP,NSTAT,1,10)
         IF (NSTAT.EQ. 0) GO TO 4
         PRINT 100,NSTAT
  100    FORMAT(*NETON FAILED,NSTAT=*,O20)
         CALL NETOFF
         STOP 111
    4    CALL NETSETP(0)
          :
    5    CALL NETCHEK
         IF (NSUP.AND.C) 2,1
          :
    1    GO TO 5
    2    CALL NETWAIT(4095,0)
C        CHECK FOR A SUP. MESSAGE OR INPUT AVAILABLE
         IF (NSUP .AND.S) 6,3
    3    IF (NSUP.AND.I) 20,5
C        SUPERVISORY MESSAGE IS AVAILABLE
    6    CALL NETGET (0,HA,TA,63)
         PFCSFC=NFETCH(TA,6LPFCSFC)
         IF (PFCSFC.EQ.CONREQ) GO TO 7
         IF (PFCSFC.EQ.FCINIT) GO TO 8
         PRINT 101,HA,TA
  101    FORMAT(*UNKNOWN SUP. MESSAGE */64(1X,O20/))
         CALL NETOFF
         STOP 222
C        CON/REQ PROCESSING
    7    ACN=NFETCH(TA,6LCONACN)
         CALL NSTORE(TA,2LRB,1)
         CALL NSTORE(TA,6LCONACT,3)
         HA=SMHDR
   40    CALL NETCHEK
         IF(NSUP.AND.C) 43,42
          :
         GO TO 40
   43    CALL NETPUT(HA,TA)
          :
         GO TO 5
C        FC/INIT PROCESSING
    8    ACN=NFETCH(TA,5LFCACN)
         CALL NSTORE(TA,2LRB,1)
         HA=SMHDR
    9    CALL NETCHEK
         IF(NSUP.AND.C) 11,10
          :
         GO TO 9
```

Enter parallel mode

Update and then check the complete bit

The complete bit is not set. Meanwhile some other processing may be performed.

The complete bit is not set. (NETWAIT is called to update the S and I bits)

Figure 6-2. Sample Program EASY in Parallel Mode (Sheet 1 of 2)

```
     11  CALL NETPUT(HA,TA)
           ⋮
         TA=10H INPUT PLS
         HA=DTHDR
         CALL NSTORE(HA,6LABHACN,ACN)
     12  CALL NETCHEK
         IF(NSUP.AND.C) 14,13
           ⋮
         GO TO 12
     13  CALL NETPUT(HA,TA)
         GO TO 5
C        INPUT IS AVAILABLE
     20  CALL NETCHEK
         IF(NSUP.AND.C) 22,21
           ⋮
     21  GO TO 20
     22  CALL NETGET(ACN,HA,TA,1)
           ⋮
         END
```

Figure 6-2. Sample Program EASY in Parallel Mode (Sheet 2 of 2)

The fragmented buffer mechanism is achieved by transmitting to NAM a vector that contains the size and address of every fragment. The format of this vector is shown in Figure 6-3. An example of the memory allocation technique is shown in figure 6-4. The following section provides a description of fragmented buffer routines.

NETGETF

The NETGETF routine inputs one data block or a supervisory message from the specified connection.

The application block header is placed in the specified header area, and the body is placed in the supplied fragmented buffer. If blocks are not available for this connection, a null block is returned, a header with an ABT of 0 is placed in the header area, and the fragmented buffer remains unchanged.

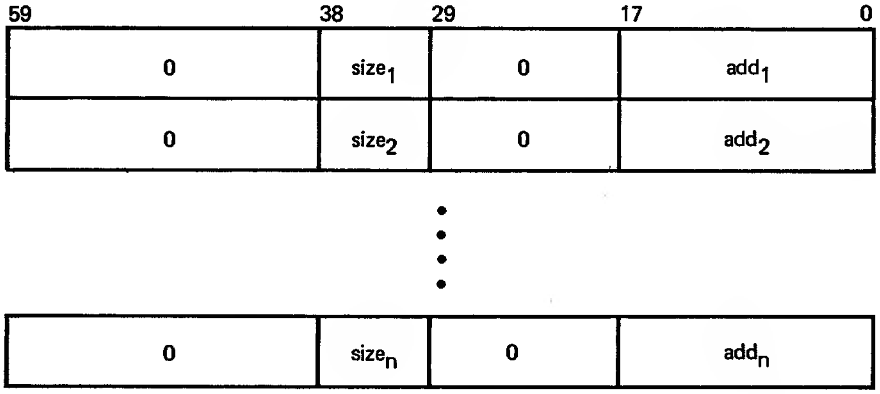
The format of NETGETF is

CALL NETGETF (acn,ha,n,va)

where:

acn indicates the ACN identifying the connection from which data is transferred (ACN of 0 is also allowed).

ha indicates header area for ABH.



59	38	29	17	0
0	$size_1$	0	add_1	
0	$size_2$	0	add_2	
⋮				
0	$size_n$	0	add_n	

$size_i$ is the length in central memory words of the i-th fragment.

add_i is the address of the i-th fragment.

Figure 6-3. Format for Fragmented Vector

n indicates the number of buffer fragments described in va (up to a maximum of 40).

va indicates the vector area containing address and size, in central memory words, of every fragment.

NETGTFL

The NETGTFL routine selects the next connection from the specified list that has input available, and inputs from it one data block or a supervisory message.

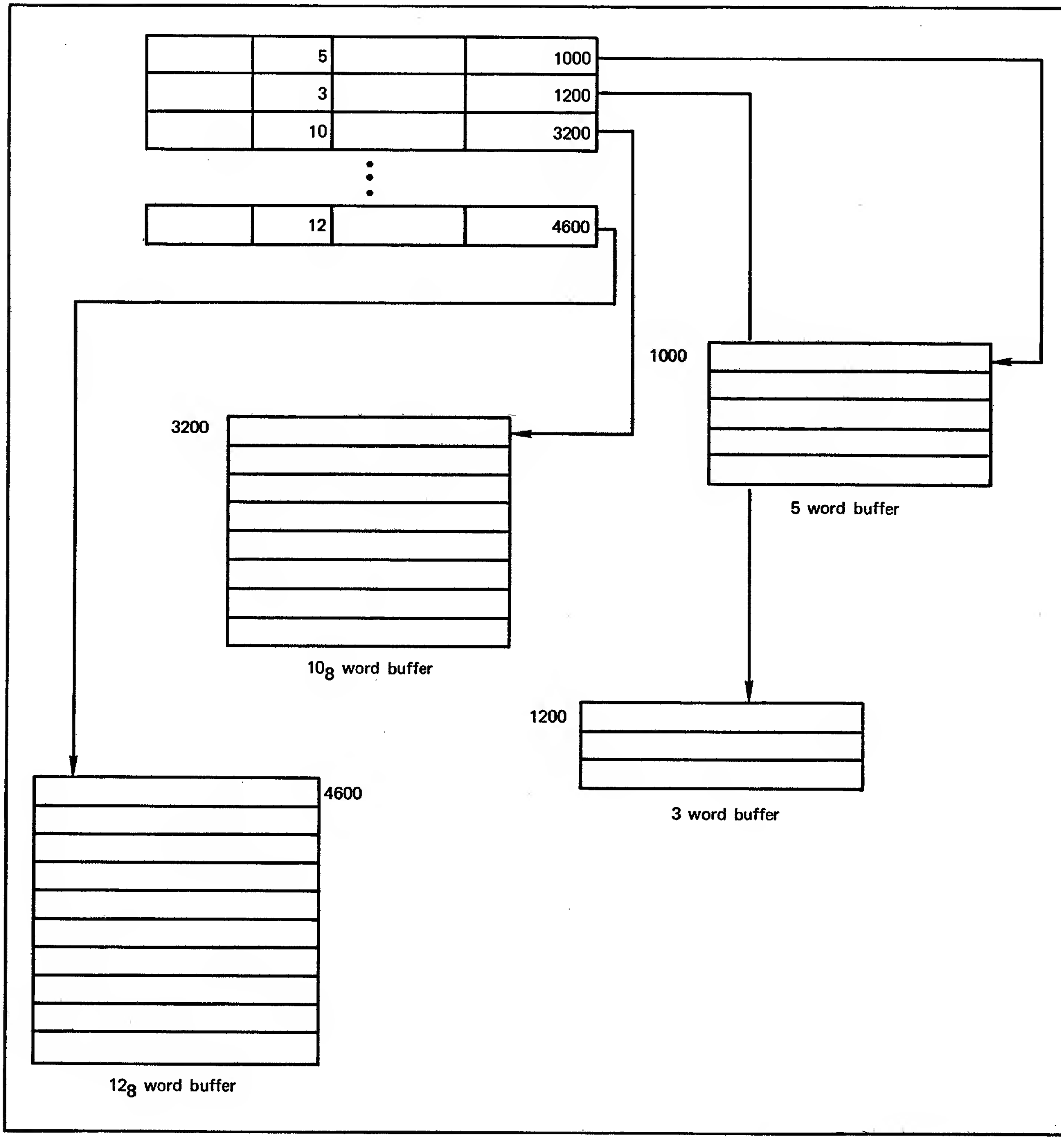


Figure 6-4. Example of Memory Allocation Technique

The header of the block is placed in the specified header area, and the body of the block in the supplied fragmented buffer. The null block is returned only if there is no input available for all the connections in the specified list. If the null block is returned, a header with an ABT of 0 is placed in the header area, and the fragmented buffer remains unchanged.

The format of NETGTFL is

```
    CALL NETGTFL (aln,ha,n,va)
```

where:

aln indicates the ALN from which connections are selected, and data is transferred.

ha,n,va serve exactly the same purpose as with NETGETF.

For all fragmented buffer input routines, if the length of the block exceeds the sum of the size values in the vector area, the block is not transmitted, but instead remains queued within NAM. The IBU bit of the ABH is set to 1, and the true length of the block is placed in the text length (TLC) field of the ABH.

NETPUTF

The NETPUTF routine outputs one data block or a supervisory message to the specified connection from the fragmented buffer area, according to the information contained in the header area.

The format of NETPUTF is

```
    CALL NETPUTF (ha,n,va)
```

where:

ha,n,va serve exactly the same purpose as with NETGETF.

APPLICATION-TO-APPLICATION COMMUNICATION

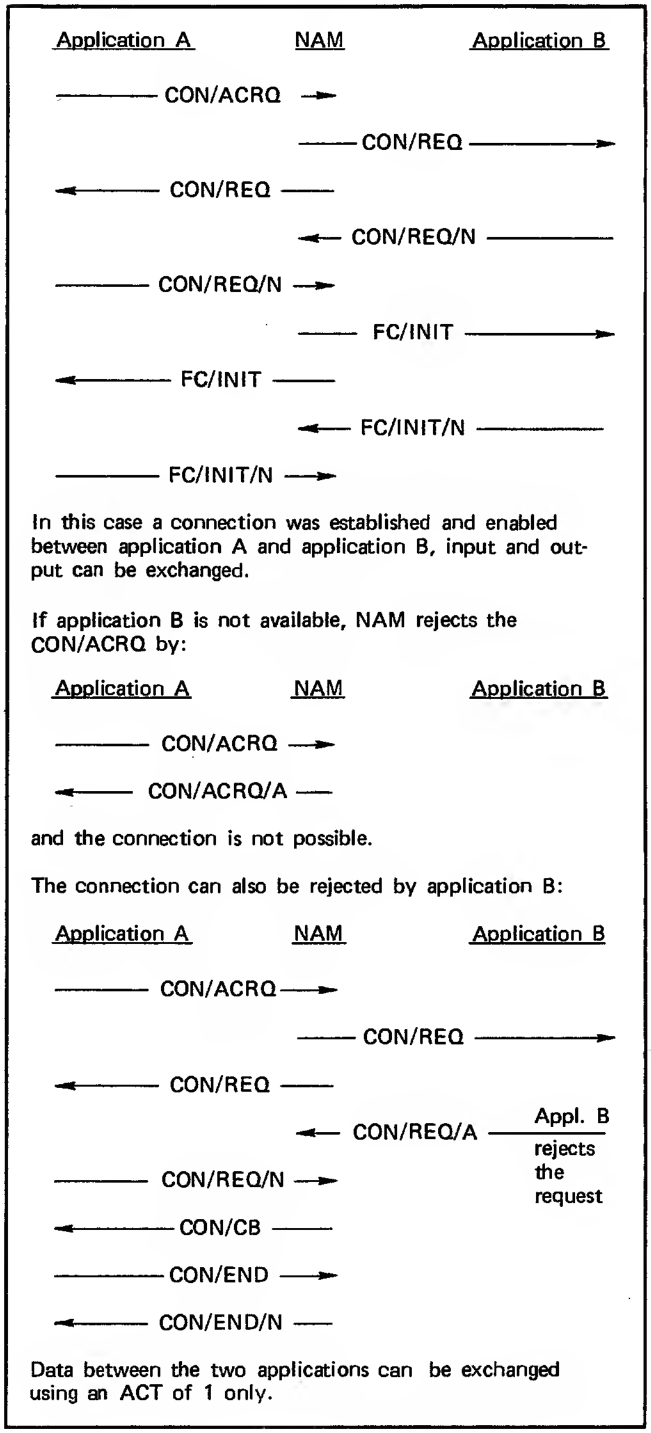
An application can connect to another application by sending NAM the CON/ACRQ SM. NAM validates the request and establishes the connection with both applications.

The supervisory message dialog shown in figure 6-5 is used when application A requires connection to application B. The format for requesting an application connection is shown in figure 6-6. If NAM can establish the connection, a CON/REQ SM is sent to both applications. If, however, the requested application is not available, an abnormal response is sent to the requesting application. See figure 6-7.

To demonstrate an application-to-application communication example, the application program MONIT is shown in figure 6-8. This program connects the ECHO program, and writes into the output file all output sent by ECHO. In order to do this, the ECHO program is slightly changed to recognize and accept application-to-application connections, and send its echoed output to the monitoring applications. It is possible that more than one application monitors ECHO. In fact, different copies of MONIT (with unique application names passed at NETON time) can concurrently run and monitor ECHO. ECHO identifies an application-to-application connection by testing the device type (DT) field of the CON/REQ SM. (Refer to figure 3-4.)

The OUTPT routine is also changed to send output to the monitoring applications which are currently connected to ECHO. The output sent by OUTPT is shown in figure 6-9.

MONIT issues a CON/ACRQ SM and waits for NAM to establish and enable the connection.



In this case a connection was established and enabled between application A and application B, input and output can be exchanged.

If application B is not available, NAM rejects the CON/ACRQ by:

and the connection is not possible.

The connection can also be rejected by application B:

Data between the two applications can be exchanged using an ACT of 1 only.

Figure 6-5. Application-to-Application Communication

59	51	50	49	43 17 0
CON	EB	RB	ACRQ	
63_{16}	0	0	02_8	

CONANM	
aname	

aname Requested application's name, one through seven alphanumeric characters with the first character being alphabetic, left-justified and blank-filled.

Figure 6-6. Format of Request Application Connection (CON/ACRQ)

59	51	50	49	43	35 17 0
CON	EB	RB	ACRQ	RC	
63_{16}	1	0	02_8	rc	

CONANM	
aname	

rc Reason code for rejecting the request:

1 = aname is not available, which might be because:

- There is no such application name in the LCF file (see appendix D)
- The requested application is not currently in the system
- The requested application has exceeded its defined connection limit

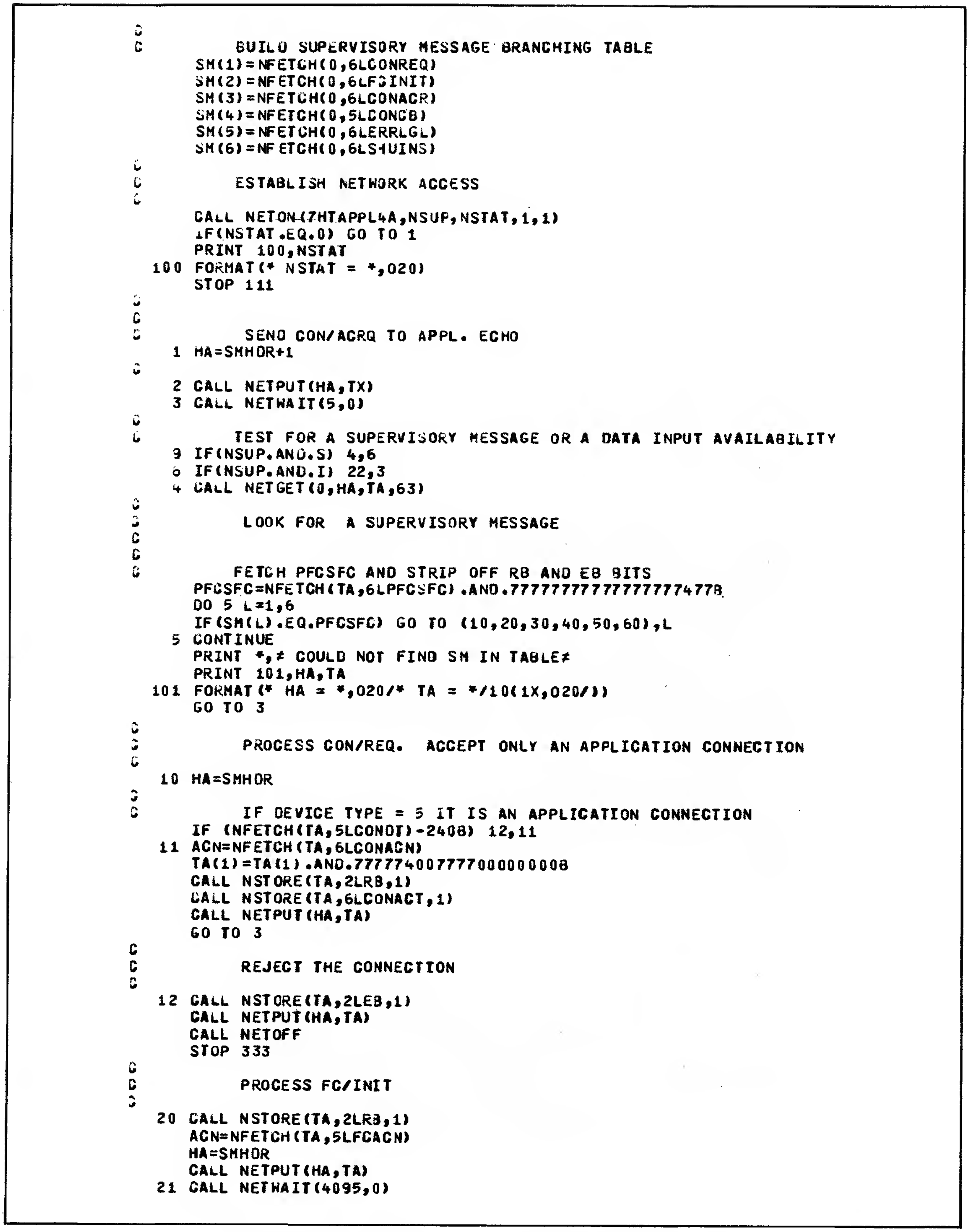
2 = Shutdown is in progress.

3 = The requesting application has exceeded its defined connection limit.

Figure 6-7. Format of Request Connection Reject (CON/ACRQ/A)

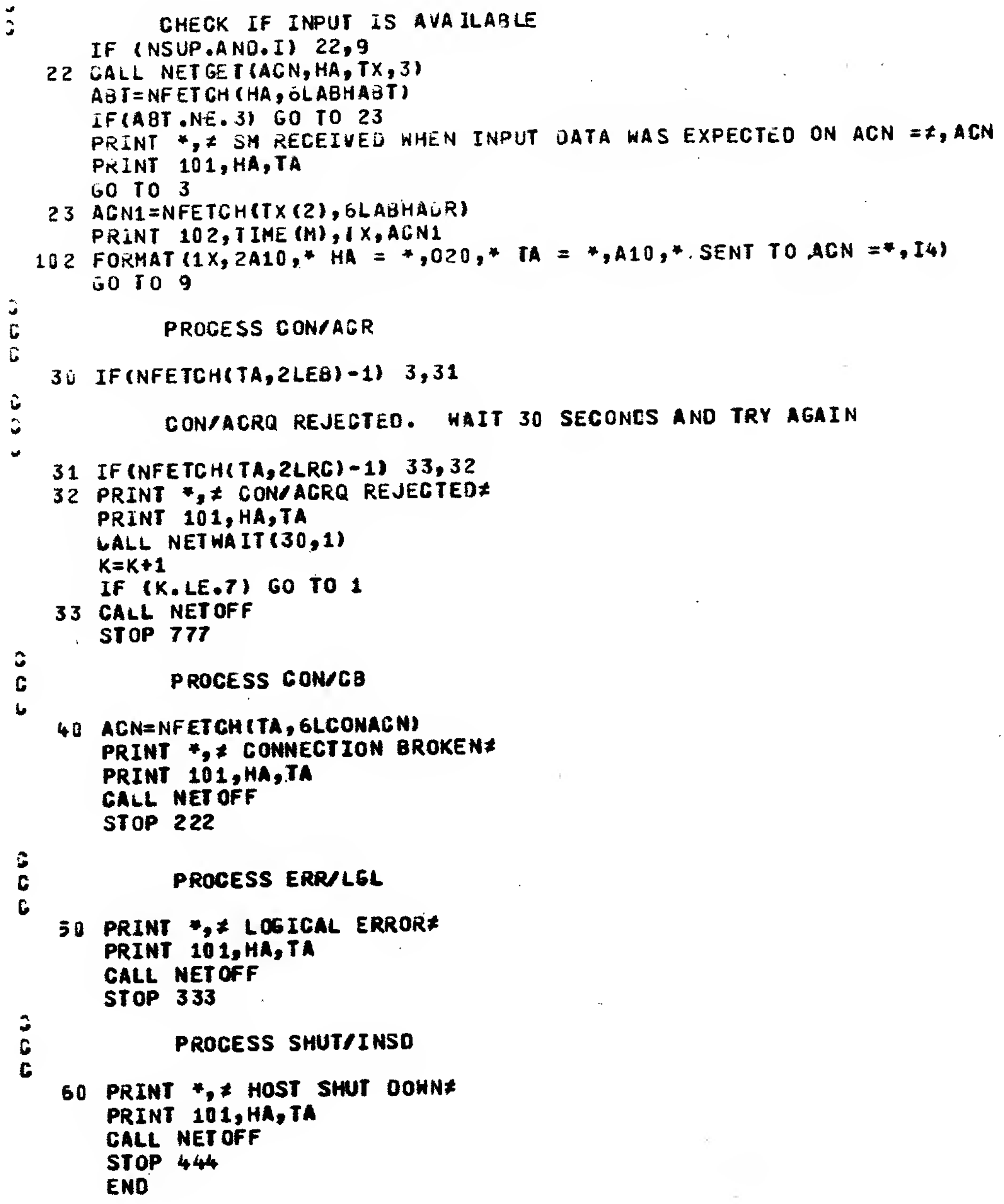
```
      PROGRAM MONIT(OUTPUT,TAPE1)
C
C         MONIT PROGRAM IS AN APPLICATION-TO-APPLICATION
C         COMMUNICATION EXAMPLE WHICH MONITORS ALL THE OUTPUT
C         SENT BY THE ECHO PROGRAM.
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION SM(6),TX(3),TA(10),XX(53)
C
C         INITIALIZE AND SET CONSTANTS
C
      K=0
      TX(1)=TX(2)=0
      SMHDR=030000000000004000018
C
C         S AND I ARE NSUP WORD MASKS
      S=SHIFT(1,55)
      I=SHIFT(1,56)
C
C         SET OUTGOING SM CONSTANTS
      CONACR=NFETCH(0,6LCONACR)
      CONEND=NFETCH(0,6LCONEND)
C
C         CONSTRUCT CON/ACRQ SM TEXT AREA
      CALL NSTORE(TX,6LPFCSFC,CONACR)
      CALL NSTORE(TX,6LCONANM,7RTAPPL4 )
```

Figure 6-8. Application-to-Application (ECHO→MONIT) (Sheet 1 of 7)



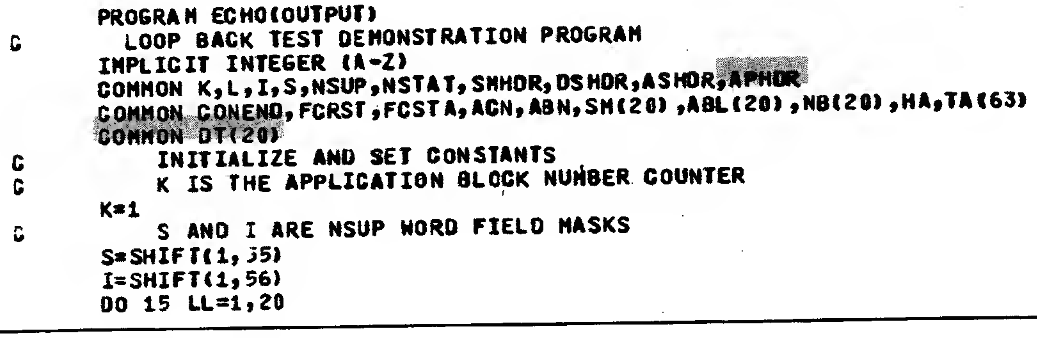
```
C
C         BUILD SUPERVISORY MESSAGE BRANCHING TABLE
      SM(1)=NFETCH(0,6LCONREQ)
      SM(2)=NFETCH(0,6LFCINIT)
      SM(3)=NFETCH(0,6LCONACR)
      SM(4)=NFETCH(0,5LCONCB)
      SM(5)=NFETCH(0,6LERRLGL)
      SM(6)=NFETCH(0,6LS4UINS)
C
C         ESTABLISH NETWORK ACCESS
C
      CALL NETON(7HTAPPL4A,NSUP,NSTAT,1,1)
      IF(NSTAT.EQ.0) GO TO 1
      PRINT 100,NSTAT
  100 FORMAT(* NSTAT = *,O20)
      STOP 111
C
C
C          SEND CON/ACRQ TO APPL. ECHO
    1 HA=SMHDR+1
C
    2 CALL NETPUT(HA,TX)
    3 CALL NETWAIT(5,0)
C
C         TEST FOR A SUPERVISORY MESSAGE OR A DATA INPUT AVAILABILITY
    9 IF(NSUP.AND.S) 4,6
    6 IF(NSUP.AND.I) 22,3
    4 CALL NETGET(0,HA,TA,63)
C
C          LOOK FOR  A SUPERVISORY MESSAGE
C
C
C         FETCH PFCSFC AND STRIP OFF RB AND EB BITS
      PFCSFC=NFETCH(TA,6LPFCSFC).AND.77777777777777777477B
      DO 5 L=1,6
      IF(SM(L).EQ.PFCSFC) GO TO (10,20,30,40,50,60),L
    5 CONTINUE
      PRINT *,≠ COULD NOT FIND SM IN TABLE≠
      PRINT 101,HA,TA
  101 FORMAT(* HA = *,O20/* TA = */10(1X,O20/))
      GO TO 3
C
C          PROCESS CON/REQ.  ACCEPT ONLY AN APPLICATION CONNECTION
C
   10 HA=SMHDR
C
C          IF DEVICE TYPE = 5 IT IS AN APPLICATION CONNECTION
      IF (NFETCH(TA,5LCONDT)-2408) 12,11
   11 ACN=NFETCH(TA,6LCONACN)
      TA(1)=TA(1).AND.777774007777000000008
      CALL NSTORE(TA,2LRB,1)
      CALL NSTORE(TA,6LCONACT,1)
      CALL NETPUT(HA,TA)
      GO TO 3
C
C          REJECT THE CONNECTION
C
   12 CALL NSTORE(TA,2LEB,1)
      CALL NETPUT(HA,TA)
      CALL NETOFF
      STOP 333
C
C          PROCESS FC/INIT
C
   20 CALL NSTORE(TA,2LRB,1)
      ACN=NFETCH(TA,5LFCACN)
      HA=SMHDR
      CALL NETPUT(HA,TA)
   21 CALL NETWAIT(4095,0)
```

Figure 6-8. Application-to-Application (ECHO→MONIT) (Sheet 2 of 7)



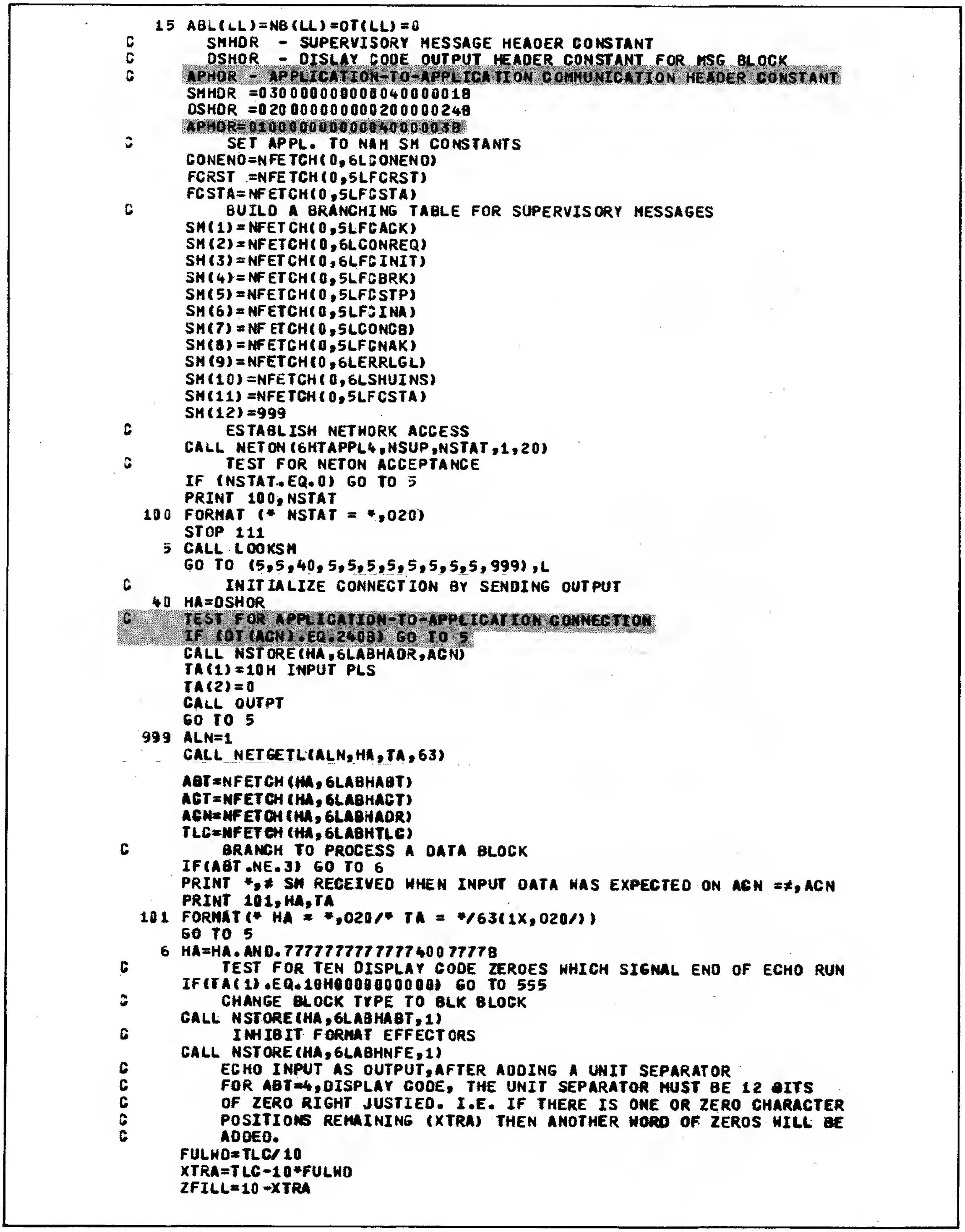
```
C
C          CHECK IF INPUT IS AVAILABLE
      IF (NSUP.AND.I) 22,9
   22 CALL NETGET(ACN,HA,TX,3)
      ABT=NFETCH(HA,6LABHABT)
      IF(ABT.NE.3) GO TO 23
      PRINT *,≠ SM RECEIVED WHEN INPUT DATA WAS EXPECTED ON ACN =≠,ACN
      PRINT 101,HA,TA
      GO TO 3
   23 ACN1=NFETCH(TX(2),6LABHADR)
      PRINT 102,TIME(M),IX,ACN1
  102 FORMAT(1X,2A10,* HA = *,O20,* TA = *,A10,*.SENT TO ACN =*,I4)
      GO TO 9
C
C          PROCESS CON/ACR
C
   30 IF(NFETCH(TA,2LEB)-1) 3,31
C
C          CON/ACRQ REJECTED.  WAIT 30 SECONDS AND TRY AGAIN
C
   31 IF(NFETCH(TA,2LRC)-1) 33,32
   32 PRINT *,≠ CON/ACRQ REJECTED≠
      PRINT 101,HA,TA
      CALL NETWAIT(30,1)
      K=K+1
      IF (K.LE.7) GO TO 1
   33 CALL NETOFF
      STOP 777
C
C          PROCESS CON/CB
C
   40 ACN=NFETCH(TA,6LCONACN)
      PRINT *,≠ CONNECTION BROKEN≠
      PRINT 101,HA,TA
      CALL NETOFF
      STOP 222
C
C          PROCESS ERR/LGL
C
   50 PRINT *,≠ LOGICAL ERROR≠
      PRINT 101,HA,TA
      CALL NETOFF
      STOP 333
C
C          PROCESS SHUT/INSD
C
   60 PRINT *,≠ HOST SHUT DOWN≠
      PRINT 101,HA,TA
      CALL NETOFF
      STOP 444
      END
```

The changes made in ECHO are shown by the shaded lines:



```
      PROGRAM ECHO(OUTPUT)
C       LOOP BACK TEST DEMONSTRATION PROGRAM
      IMPLICIT INTEGER (A-Z)
      COMMON K,L,I,S,NSUP,NSTAT,SMHDR,DSHDR,ASHDR,APHDR
      COMMON CONEND,FCRST,FCSTA,ACN,ABN,SM(20),ABL(20),NB(20),HA,TA(63)
      COMMON DT(20)
C          INITIALIZE AND SET CONSTANTS
C          K IS THE APPLICATION BLOCK NUMBER COUNTER
      K=1
C          S AND I ARE NSUP WORD FIELD MASKS
      S=SHIFT(1,55)
      I=SHIFT(1,56)
      DO 15 LL=1,20
```

Figure 6-8. Application-to-Application (ECHO→MONIT) (Sheet 3 of 7)



```
   15 ABL(LL)=NB(LL)=OT(LL)=0
C        SMHDR  - SUPERVISORY MESSAGE HEADER CONSTANT
C        DSHDR  - DISLAY CODE OUTPUT HEADER CONSTANT FOR MSG BLOCK
C     APHDR - APPLICATION-TO-APPLICATION COMMUNICATION HEADER CONSTANT
      SMHDR =0300000000004000001B
      DSHDR =0200000000002000024B
      APHDR=0100000000004000003B
C          SET APPL. TO NAM SM CONSTANTS
      CONENO=NFETCH(0,6LCONEND)
      FCRST =NFETCH(0,5LFCRST)
      FCSTA=NFETCH(0,5LFCSTA)
C          BUILD A BRANCHING TABLE FOR SUPERVISORY MESSAGES
      SM(1)=NFETCH(0,5LFCACK)
      SM(2)=NFETCH(0,6LCONREQ)
      SM(3)=NFETCH(0,6LFCINIT)
      SM(4)=NFETCH(0,5LFCBRK)
      SM(5)=NFETCH(0,5LFCSTP)
      SM(6)=NFETCH(0,5LFCINA)
      SM(7)=NFETCH(0,5LCONCB)
      SM(8)=NFETCH(0,5LFCNAK)
      SM(9)=NFETCH(0,6LERRLGL)
      SM(10)=NFETCH(0,6LSHUINS)
      SM(11)=NFETCH(0,5LFCSTA)
      SM(12)=999
C          ESTABLISH NETWORK ACCESS
      CALL NETON(6HTAPPL4,NSUP,NSTAT,1,20)
C          TEST FOR NETON ACCEPTANCE
      IF (NSTAT.EQ.0) GO TO 5
      PRINT 100,NSTAT
  100 FORMAT (* NSTAT = *,O20)
      STOP 111
    5 CALL LOOKSM
      GO TO (5,5,40,5,5,5,5,5,5,5,5,999),L
C          INITIALIZE CONNECTION BY SENDING OUTPUT
   40 HA=DSHDR
C     TEST FOR APPLICATION-TO-APPLICATION CONNECTION
      IF (DT(ACN).EQ.240B) GO TO 5
      CALL NSTORE(HA,6LABHADR,ACN)
      TA(1)=10H INPUT PLS
      TA(2)=0
      CALL OUTPT
      GO TO 5
  999 ALN=1
      CALL NETGETL(ALN,HA,TA,63)

      ABT=NFETCH(HA,6LABHABT)
      ACT=NFETCH(HA,6LABHACT)
      ACN=NFETCH(HA,6LABHADR)
      TLC=NFETCH(HA,6LABHTLC)
C          BRANCH TO PROCESS A DATA BLOCK
      IF(ABT.NE.3) GO TO 6
      PRINT *,≠ SM RECEIVED WHEN INPUT DATA WAS EXPECTED ON ACN =≠,ACN
      PRINT 101,HA,TA
  101 FORMAT(* HA = *,O20/* TA = */63(1X,O20/))
      GO TO 5
    6 HA=HA.AND.777777777777740077778
C          TEST FOR TEN DISPLAY CODE ZEROES WHICH SIGNAL END OF ECHO RUN
      IF(TA(1).EQ.10H0000000000) GO TO 555
C          CHANGE BLOCK TYPE TO BLK BLOCK
      CALL NSTORE(HA,6LABHABT,1)
C           INHIBIT FORMAT EFFECTORS
      CALL NSTORE(HA,6LABHNFE,1)
C          ECHO INPUT AS OUTPUT,AFTER ADDING A UNIT SEPARATOR
C          FOR ABT=4,DISPLAY CODE, THE UNIT SEPARATOR MUST BE 12 BITS
C          OF ZERO RIGHT JUSTIED. I.E. IF THERE IS ONE OR ZERO CHARACTER
C          POSITIONS REMAINING (XTRA) THEN ANOTHER WORD OF ZEROS WILL BE
C          ADDED.
      FULWD=TLC/10
      XTRA=TLC-10*FULWD
      ZFILL=10-XTRA
```

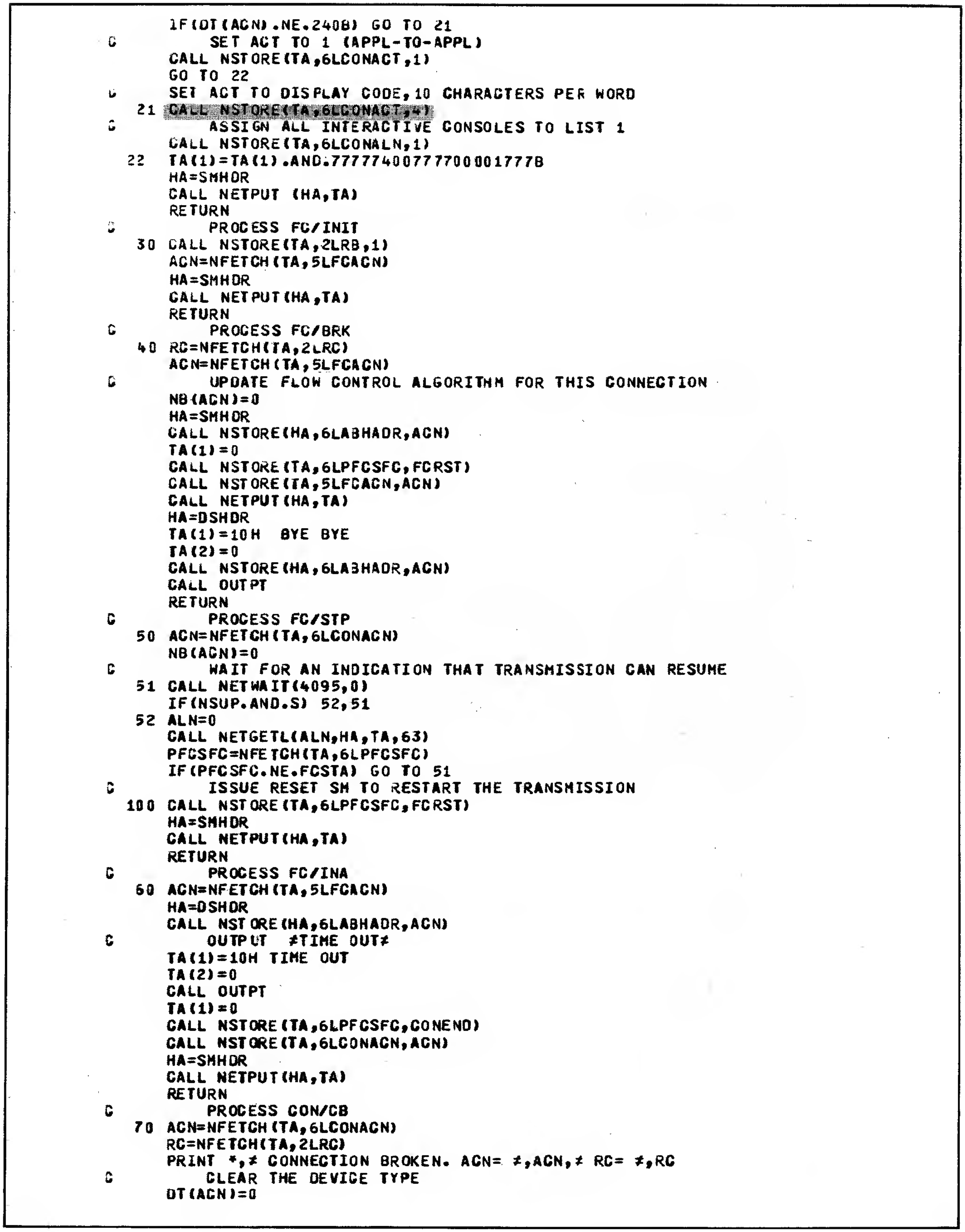
Figure 6-8. Application-to-Application (ECHO→MONIT) (Sheet 4 of 7)

```
      TLC=TLC+ZFILL
      MASK=77000000000000000000B
      IF(XTRA.EQ.0)TA(FULWD+1)=0
      TA(FULWD+1)=TA(FULWD+1).AND.SHIFT(MASK,-(6*XTRA))
      IF(ZFILL.NE.1) GO TO 71
      TLC=TLC+10
      TA(FULWD+2)=0
   71 CALL NSTORE(HA,6LABHTLC,TLC)
      CALL OUTPT
      CALL LOOKSM
      GO TO (40,5,40,5,5,5,5,5,5,5,5,999),L
C          SEND CON/END TO ALL ACTIVE CONNECTIONS
  555 TA(1)=0
      CALL NSTORE(TA,6LPFCSFC,CONEND)
      HA=SMHDR
      DO 556 LL=1,20
      IF(ABL(LL).EQ.0) GO TO 556
      CALL NSTORE(TA,6LCONACN,LL)
      CALL NETPUT(HA,TA)
  556 CONTINUE
      CALL NETOFF
      STOP 777
      END
```

The LOOKSM changes are shown by the shaded lines:

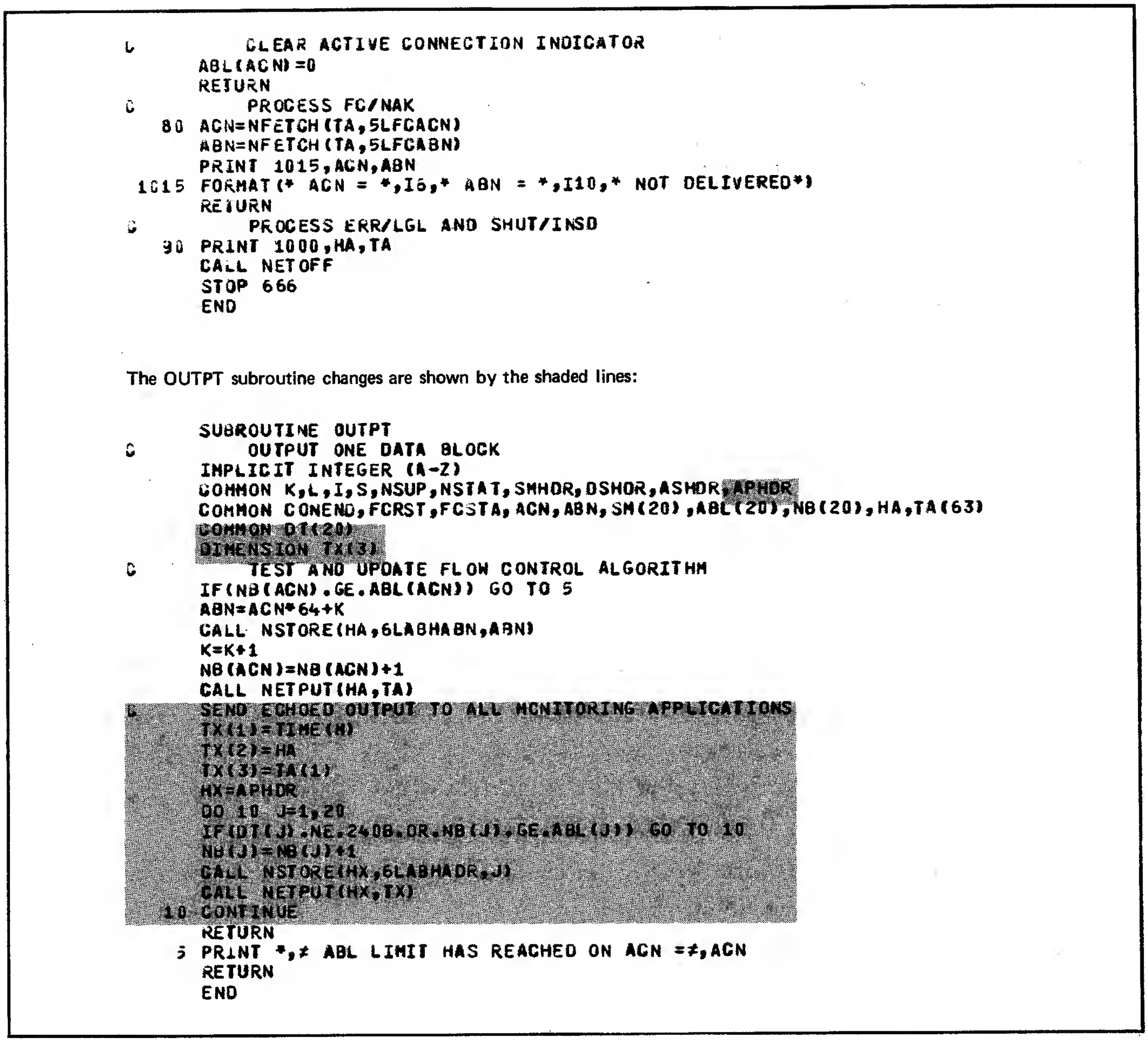
```
      SUBROUTINE LOOKSM
C          PROCESS ASYNCHRONOUS SUPERVISORY MESSAGES
      IMPLICIT INTEGER (A-Z)
      COMMON K,L,I,S,NSUP,NSTAT,SMHDR,OSHDR,ASHDR,APHDR
      COMMON CONEND,FCRST,FCSTA,ACN,ABN,SM(20),ABL(20),NB(20),HA,TA(63)
      COMMON DT(20)
      L=12
C          CHECK FOR OUTSTANDING SUPERVISORY MESSAGES
      IF(NSUP.AND.S) 6,3
C          PAUSE FOR INPUT DATA OR A SUPERVISORY MESSAGE
    3 CALL NETWAIT(4095,0)
      IF (NSUP.AND.S) 6,5
    5 IF(NSUP.AND.I) 2,3
    2 RETURN
    6 ALN=0
C          FETCH AN ASYNCHRONOUS SM FROM ACN=0
      CALL NETGETL(ALN,HA,TA,63)
C          FETCH PFCSFC AND STRIP OFF RB AND EB BITS
      PFCSFC=NFETCH(TA,6LPFCSFC).AND.777777777777777477B
      DO 8 L=1,20
      IF(SM(L).EQ.999) GO TO 9
C          BRANCH ACCORDING SUPERVISORY MESSAGE CODE
      IF(PFCSFC.EQ.SM(L)) GO TO(10,20,30,40,50,60,70,80,90,90,100),L
    8 CONTINUE
    9 PRINT *,≠ COULD NOT FIND SM IN TABLE≠
      PRINT 1000,HA,TA
 1000 FORMAT(* HA =*,020/* TA = */63(1X,020/))
      GO TO 3
C          PROCESS FC/ACK
   10 ACN=NFETCH(TA,5LFCACN)
C          UPDATE FLOW CONTROL ALGORITHM
      NB(ACN)=NB(ACN)-1
      RETURN
C          PROCESS CON/REQ
   20 ACN=NFETCH(TA,6LCONACN)
      ABL(ACN)=NFETCH(TA,6LCONABL)
      NB(ACN)=0
C     STORE DEVICE TYPE (FOR IDENTIFYING LATER AN APP-TO-APP CONNECTION)
      DT(ACN)=NFETCH(TA,5LCONDT)
C          SET RES ONSE BIT TO ACCEPT THE CONNECTION
      CALL NSTORE(TA,2LRB,1)
C          CHECK IF APPL-TO-APPL CONNECTION
```

Figure 6-8. Application-to-Application (ECHO→MONIT) (Sheet 5 of 7)



```
      IF(DT(ACN).NE.240B) GO TO 21
C         SET ACT TO 1 (APPL-TO-APPL)
      CALL NSTORE(TA,6LCONACT,1)
      GO TO 22
C     SET ACT TO DISPLAY CODE,10 CHARACTERS PER WORD
   21 CALL NSTORE(TA,6LCONACT,4)
C         ASSIGN ALL INTERACTIVE CONSOLES TO LIST 1
      CALL NSTORE(TA,6LCONALN,1)
  22  TA(1)=TA(1).AND.77777400777700001777B
      HA=SMHDR
      CALL NETPUT (HA,TA)
      RETURN
C         PROCESS FC/INIT
   30 CALL NSTORE(TA,2LRB,1)
      ACN=NFETCH(TA,5LFCACN)
      HA=SMHDR
      CALL NETPUT(HA,TA)
      RETURN
C         PROCESS FC/BRK
   40 RC=NFETCH(TA,2LRC)
      ACN=NFETCH(TA,5LFCACN)
C         UPDATE FLOW CONTROL ALGORITHM FOR THIS CONNECTION
      NB(ACN)=0
      HA=SMHDR
      CALL NSTORE(HA,6LABHADR,ACN)
      TA(1)=0
      CALL NSTORE(TA,6LPFCSFC,FCRST)
      CALL NSTORE(TA,5LFCACN,ACN)
      CALL NETPUT(HA,TA)
      HA=DSHDR
      TA(1)=10H  BYE BYE
      TA(2)=0
      CALL NSTORE(HA,6LABHADR,ACN)
      CALL OUTPT
      RETURN
C         PROCESS FC/STP
   50 ACN=NFETCH(TA,6LCONACN)
      NB(ACN)=0
C         WAIT FOR AN INDICATION THAT TRANSMISSION CAN RESUME
   51 CALL NETWAIT(4095,0)
      IF(NSUP.AND.S) 52,51
   52 ALN=0
      CALL NETGETL(ALN,HA,TA,63)
      PFCSFC=NFETCH(TA,6LPFCSFC)
      IF(PFCSFC.NE.FCSTA) GO TO 51
C         ISSUE RESET SM TO RESTART THE TRANSMISSION
  100 CALL NSTORE(TA,6LPFCSFC,FCRST)
      HA=SMHDR
      CALL NETPUT(HA,TA)
      RETURN
C         PROCESS FC/INA
   60 ACN=NFETCH(TA,5LFCACN)
      HA=DSHDR
      CALL NSTORE(HA,6LABHADR,ACN)
C         OUTPUT  ≠TIME OUT≠
      TA(1)=10H TIME OUT
      TA(2)=0
      CALL OUTPT
      TA(1)=0
      CALL NSTORE(TA,6LPFCSFC,CONEND)
      CALL NSTORE(TA,6LCONACN,ACN)
      HA=SMHDR
      CALL NETPUT(HA,TA)
      RETURN
C         PROCESS CON/CB
   70 ACN=NFETCH(TA,6LCONACN)
      RC=NFETCH(TA,2LRC)
      PRINT *,≠ CONNECTION BROKEN. ACN= ≠,ACN,≠ RC= ≠,RC
C         CLEAR THE DEVICE TYPE
      DT(ACN)=0
```

Figure 6-8. Application-to-Application (ECHO→MONIT) (Sheet 6 of 7)



```
C         CLEAR ACTIVE CONNECTION INDICATOR
      ABL(ACN)=0
      RETURN
C         PROCESS FC/NAK
   80 ACN=NFETCH(TA,5LFCACN)
      ABN=NFETCH(TA,5LFCABN)
      PRINT 1015,ACN,ABN
 1015 FORMAT(* ACN = *,I6,* ABN = *,I10,* NOT DELIVERED*)
      RETURN
C         PROCESS ERR/LGL AND SHUT/INSD
   90 PRINT 1000,HA,TA
      CALL NETOFF
      STOP 666
      END
```

The OUTPT subroutine changes are shown by the shaded lines:

```
      SUBROUTINE OUTPT
C         OUTPUT ONE DATA BLOCK
      IMPLICIT INTEGER (A-Z)
      COMMON K,L,I,S,NSUP,NSTAT,SMHDR,DSHDR,ASHDR,APHDR
      COMMON CONEND,FCRST,FCSTA,ACN,ABN,SM(20),ABL(20),NB(20),HA,TA(63)
      COMMON DT(20)
      DIMENSION TX(3)
C         TEST AND UPDATE FLOW CONTROL ALGORITHM
      IF(NB(ACN).GE.ABL(ACN)) GO TO 5
      ABN=ACN*64+K
      CALL NSTORE(HA,6LABHABN,ABN)
      K=K+1
      NB(ACN)=NB(ACN)+1
      CALL NETPUT(HA,TA)
C     SEND ECHOED OUTPUT TO ALL MCNITORING APPLICATIONS
      TX(1)=TIME(H)
      TX(2)=HA
      TX(3)=TA(1)
      HX=APHDR
      DO 10 J=1,20
      IF(DT(J).NE.240B.OR.NB(J).GE.ABL(J)) GO TO 10
      NB(J)=NB(J)+1
      CALL NSTORE(HX,6LABHADR,J)
      CALL NETPUT(HX,TX)
   10 CONTINUE
      RETURN
    5 PRINT *,≠ ABL LIMIT HAS REACHED ON ACN =≠,ACN
      RETURN
      END
```

Figure 6-8. Application-to-Application (ECHO→MONIT) (Sheet 7 of 7)

hh.mm.ss	ABH	ccc....c
word 1	word 2	word 3

hh.mm.ss The time this message was sent to a terminal.

ABH The original ABH sent to a terminal.

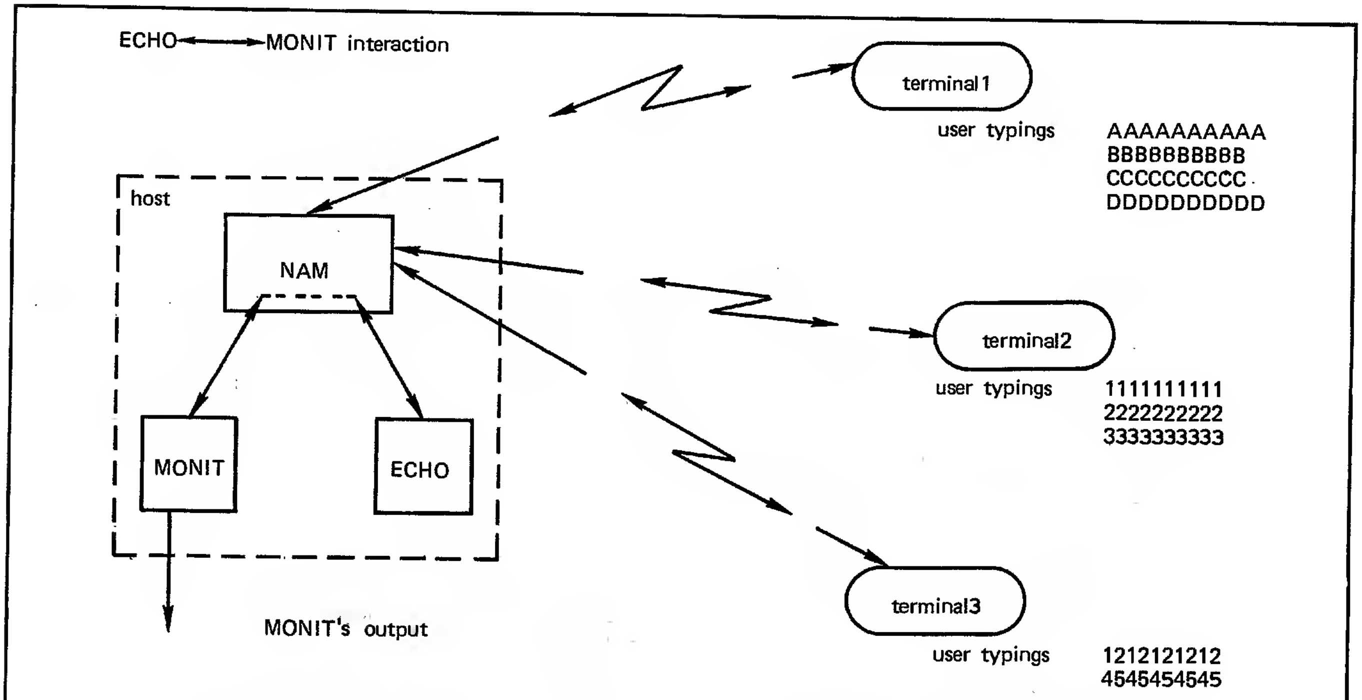
ccc....c The echoed characters sent to a terminal.

Figure 6-9. Output Sent by OUTPT

If CON/ACRQ is rejected, MONIT checks the reason code (RC). If RC is 1 (for example, ECHO is probably not currently in the system), MONIT reissues the request again in 30-second intervals, up to seven times. If RC is not equal to 1 or 7, CON/ACRQ SMs are rejected and MONIT NETOFFs. If the connection with ECHO is enabled, MONIT NETGETs the messages sent to it by ECHO, and outputs them prefixed by the time they were accepted. See figure 6-10.

THE DEBUG OPTION

NAM provides debugging options used for program tracing and verification. This feature can be turned on and off at execution time, and can log supervisory message dialogs and data message transactions between NAM and the application.



```
18.05.44  18.05.36  HA = 02000100000020000012 TA = INPUT PLS  SENT TO ACN =  1
18.06.12  18.06.06  HA = 01000100000020000012 TA = AAAAAAAAAA SENT TO ACN =  1
18.06.14  18.06.07  HA = 02000100000020000012 TA = INPUT PLS  SENT TO ACN =  1
18.06.55  18.06.50  HA = 01000100000020000012 TA = BBBBBBBBBB SENT TO ACN =  1
18.07.01  18.06.51  HA = 02000100000020000012 TA = INPUT PLS  SENT TO ACN =  1
18.07.46  18.07.45  HA = 02000200000020000012 TA = INPUT PLS  SENT TO ACN =  2
18.08.32  18.08.31  HA = 01000200000020000012 TA = 1111111111 SENT TO ACN =  2
18.03.33  18.08.31  HA = 02000200000020000012 TA = INPUT PLS  SENT TO ACN =  2
18.09.25  18.09.25  HA = 01000100000020000012 TA = CCCCCCCCCC SENT TO ACN =  1
18.09.26  18.09.25  HA = 02000100000020000012 TA = INPUT PLS  SENT TO ACN =  1
18.09.51  18.09.48  HA = 01000200000020000012 TA = 2222222222 SENT TO ACN =  2
18.09.52  18.09.48  HA = 02000200000020000012 TA = UNPUT PLS  SENT TO ACN =  2
18.11.12  18.11.11  HA = 02000300000020000012 TA = INPUT PLS  SENT TO ACN =  3
18.11.54  18.11.54  HA = 01000200000020000012 TA = 3333333333 SENT TO ACN =  2
18.11.55  18.11.54  HA = 02000200000020000012 TA = INPUT PLS  SENT TO ACN =  2
18.12.43  18.12.43  HA = 01000300000020000012 TA = 1212121212 SENT TO ACN =  3
18.12.45  18.12.44  HA = 02000300000020000012 TA = GNPUT PLS  SENT TO ACN =  3
18.13.46  18.13.45  HA = 01000300000020000012 TA = 4545454545 SENT TO ACN =  3
18.13.47  18.13.45  HA = 02000300000020000012 TA = INPUT PLS  SENT TO ACN =  3
18.14.14  18.14.13  HA = 01000100000020000012 TA = DDDDDDDDDD SENT TO ACN =  1
18.14.16  18.14.14  HA = 02000100000020000012 TA = INPUT PLS  SENT TO ACN =  1
```

The time when the message was accepted by MONIT

The time the message was sent by ECHO

Original ABH sent by ECHO

Echoed output

The ACN to which the message was sent

Figure 6-10. ECHO-MONIT Interaction

The system library, NETIO, contains all AIP procedures without the debugging option, while system library, NETIOD, contains AIP procedures with the debugging options.

If debugging options are not required, load AIP procedures from NETIO. Loading from NETIOD requires additional central memory at the control point of the application.

In order to check if the debugging options are available and turn them on or off, the AIP NETDBG routine must be called:

CALL NETDBG (opt_1,opt_2,stat)

where:

opt_1 = 0 turns on logging of asynchronous supervisory messages (without FC/ACK).

≠ 0 turns off logging of asynchronous supervisory messages.

opt_2 = 0 turns on logging of data messages (including FC/ACK) up to a maximum of 10 words.

≠ 0 turns off logging of data messages.

stat = 0 indicates debugging options are available (AIP procedures were loaded from NETIOD).

≠ 1 indicates debugging options are not available (AIP procedures were loaded from NETIO).

All debug output is written to the local file ZZZZZDN. Data is formatted as display code character strings which can later be rewound and copied to the OUTPUT file routed to the print queue. Debug output is written to the log file by calling NETON, NETOFF, NETDBG and each one of AIP input/output routines. NETON and NETOFF calls are logged to indicate the start and end of the NAM interface. The NETDBG call is logged to indicate the debugging options selected. The AIP input/output routines are logged to indicate supervisory messages and the data messages exchanged between NAM and the application. Table 6-1 summarizes the output information and the conditions under which it is logged. A debug output example is shown in figure 6-11.

TABLE 6-1. OUTPUT INFORMATION AND CONDITIONS OF DEBUG OPTIONS

Routine Called	Information Logged	Notes
NETON	NETON parameters, date and time, routine name and address	Logged only if successful status = 0; refer to section 3.
NETOFF	Date and time, routine name and address	An EOR is written after NETOFF call is logged.
NETDBG	Debugging options selected, time, routine name and address	
NETGET NETGETL NETGETF NETGTFL NETPUT NETPUTF	Header and routine parameters, text area and/or fragmented area contents, time, routine name and address	Logged only if the call results in the transfer of a message, and the appropriate option is on.

THE STATISTICAL OPTION

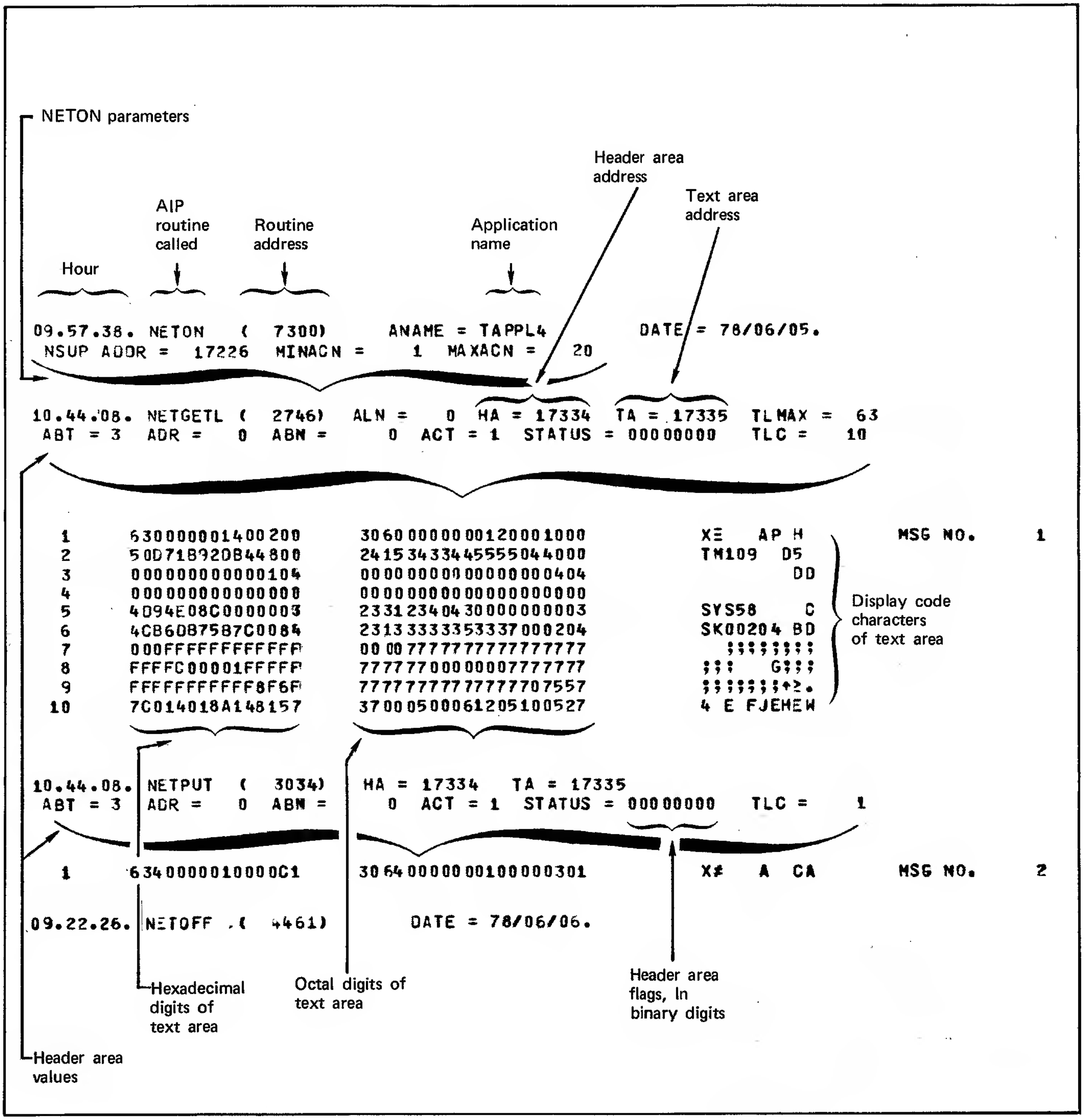
Statistical information can be gathered by NAM from the local file ZZZZZSN. The statistical option is automatically activated when loading from the NETIOD system library. In order to check option availability and turn it on or off, the user must call NETSTC as follows:

CALL NETSTC (opt,stat)

where:

opt = 0 turns statistical option on.

= 1 turns statistical option off.


NETON parameters
Header area address
Text area address
AIP routine called
Routine address
Application name
Hour
Display code characters of text area
Hexadecimal digits of text area
Octal digits of text area
Header area flags, In binary digits
Header area values

```
09.57.38. NETON    (  7300)     ANAME = TAPPL4          DATE = 78/06/05.
 NSUP ADDR =  17226  MINACN =     1  MAXACN =    20

10.44.08. NETGETL (  2746)  ALN =    0  HA = 17334  TA = 17335  TLMAX =   63
 ABT = 3  ADR =   0  ABN =        0  ACT = 1  STATUS = 00000000  TLC =    10

   1     630000001400200      30600000000120001000          X≡   AP H          MSG NO.      1
   2     50D71B920B44800      24153433445555044000          TM109  D5
   3     000000000000104      00000000000000000404                DD
   4     000000000000000      00000000000000000000
   5     4094E08C0000003      23312340430000000003          SYS58     C
   6     4CB6087587C0084      23133333353337000204          SK00204 BD
   7     000FFFFFFFFFFFF      00007777777777777777             ;;;;;;;;
   8     FFFFC00001FFFFF      77777700000007777777          ;;;    G;;;
   9     FFFFFFFFFFF8F6F      77777777777777707557          ;;;;;;;↑≥.
  10     7C014018A148157      37000500061205100527          4 E FJEHEW

10.44.08. NETPUT   (   3034)   HA = 17334   TA = 17335
 ABT = 3  ADR =    0  ABN =     0  ACT = 1  STATUS = 00000000   TLC =     1

   1     6340000010000C1      30640000000100000301          X≠    A  CA          MSG NO.      2

09.22.26. NETOFF .(  4461)        DATE = 78/06/06.
```

Figure 6-11. Debug Output Example

stat = 0 indicates statistical option is available (AIP procedures were loaded from NETIOD).

= 1 indicates statistical option is not available (AIP procedures were loaded from NETIO).

The statistical information written to ZZZZZSN is shown in figure 6-12.

```
    NAM STATISTICS GATHERING TERMINATED
    NETyyy   DATE xx/xx/xx   TIME xx.xx.xx

where yyy is either OFF or STC.

    CPU TIME USED:  xxxxxx SEC
    FREQUENCY OF PROCEDURE CALLS

        NETCHEK    xxxxxx
        NETGET     xxxxxx
        NETGETF    xxxxxx
        NETGETL    xxxxxx
        NETGTFL    xxxxxx
        NETPUT     xxxxxx
        NETPUTF    xxxxxx
        NETSETP    xxxxxx
        NETWAIT    xxxxxx

    NUMBER OF WORKLIST TRANSFER ATTEMPTS

        SUCCESSFUL      xxxxxx
        UNSUCCESSFUL    xxxxxx

    NUMBER OF INPUT/OUTPUT BLOCKS
    TRANSFERRED

        INPUT     ABT=0     xxxxxx
        INPUT     ABT=1     xxxxxx
        INPUT     ABT=2     xxxxxx
        INPUT     ABT=3     xxxxxx

        OUTPUT    ABT=1     xxxxxx
        OUTPUT    ABT=2     xxxxxx
        OUTPUT    ABT=3     xxxxxx

    NUMBER OF ERRORS

        LOGICAL ERROR       xxxxxx
          NAK-S             xxxxxx

where xxxxxx is a 60-bit signed integer, if 0 the corre-
sponding line is not printed.
```

Figure 6-12. NAM Statistic Gathering

Between the STATISTICS GATHERING STARTED and TERMINATED messages, if one of the counters overflows, the corresponding line of statistical information is written to file ZZZZZSN, preceded by the message

****COUNTER OVERFLOW****

and the corresponding counter is reset to 0. See figure 6-13.

```
NAM STATISTICS GATHERING STARTED
NETON   DATE 77/09/27.   TIME 04.11.09.

NAM STATISTICS GATHERING TERMINATED
NETOFF   DATE 77/09/27.   TIME 04.11.09.

CPU TIME USED:                      0.311 SEC

NUMBER OF PROCEDURE CALLS
    NETGTFL                             3
    NETPUT                              5
    NETPUTF                             2

NUMBER OF WORKLIST TRANSFER ATTEMPTS

NUMBER OF INPUT/OUTPUT BLOCKS TRANSFERRED
    INPUT     ABT=0                     3
    OUTPUT    ABT=1                     4
    OUTPUT    ABT=2                     1

NUMBER OF ERRORS
```

Figure 6-13. Statistical Option Output Example

OPERATING SYSTEM CHARACTER SETS

CDC operating systems offer the following variations of a basic character set:

CDC 63-character set

CDC 64-character set

ASCII 63-character set

ASCII 64-character set

ASCII 128-character set

The set in use at a particular installation is specified when the operating system is installed or deadstarted. Depending on another installation option, the system assumes an input deck has been punched either in 026 or 029 mode (regardless of the character set in use).

For card decks read at local peripherals, alternate keypunch modes can be specified by a 26 or 29 punched in columns 79 and 80 of any 6/7/9 card or 7/8/9 card. The specified mode remains in effect through the end of the job unless it is reset by specification of the alternate mode on a subsequent 7/8/9 card or 6/7/9 card.

For card decks read from remote batch stations through RBF, the ability to use the alternate mode selection is dependent upon the remote terminal equipment. See Input Deck Structure in the RBF reference manual.

Graphic character representation appearing at a terminal or printer depends on the installation character set and the terminal type. Characters shown in the CDC Graphic column of the standard character set table (table A-1) are applicable to BCD terminals; ASCII graphic characters are applicable to ASCII terminals. Tables A-1 through A-4 are provided for the reader's use while coding an application program to run under the operating system. They do not describe character transmissions between an application program and the network. The BCD code shown in table A-1 is a 7-track tape code, while the ASCII code shown is a 9-track tape code. Network character translation is described in the following subsection.

128-CHARACTER ASCII SET

Table A-5 contains the 128-character ASCII set supported by the Network Access Method. A 96-character subset consists of the rightmost six columns; a 64-character subset consists of the middle four columns. Note that display code equivalents exist for the characters in this 64-character subset only.

While the network supports the 128-character set, terminal restrictions might require that output to a device be limited to a smaller subset. This is accomplished by replacing the control characters in columns 0 and 1 of table A-5 with blanks to produce the 96-character subset, and additionally replacing the characters in columns 6 and 7 with the corresponding characters from columns 4 and 5, respectively, to produce the 64-character subset.

Similarly, input from a device may be limited to a smaller subset by the device itself because of an inability to produce the full 128-character set. A character input from a device using a character set other than ASCII is converted to an equivalent ASCII character; characters without ASCII character equivalents are replaced by the ASCII blank character.

An application can also cause character replacement as described for output above as well as the character conversion, by requesting display-coded input from the network.

The 7-bit hexadecimal code value for each character consists of the character's column number in the table, followed by its row number. For example, N is in row E and column 4, so its value is $4E_{16}$.

TABLE A-1. STANDARD CHARACTER SETS

Display Code (octal)	CDC			ASCII		
	Graphic	Hollerith Punch (026)	External BCD Code	Graphic Subset	Punch (029)	Code (octal)
00[†]	: (colon)[††]	8-2	00	: (colon)[††]	8-2	072
01	A	12-1	61	A	12-1	101
02	8	12-2	62	B	12-2	102
03	C	12-3	63	C	12-3	103
04	D	12-4	64	D	12-4	104
05	E	12-5	65	E	12-5	105
06	F	12-6	66	F	12-6	106
07	G	12-7	67	G	12-7	107
10	H	12-8	70	H	12-8	110
11	I	12-9	71	I	12-9	111
12	J	11-1	41	J	11-1	112
13	K	11-2	42	K	11-2	113
14	L	11-3	43	L	11-3	114
15	M	11-4	44	M	11-4	115
16	N	11-5	45	N	11-5	116
17	O	11-6	46	O	11-6	117
20	P	11-7	47	P	11-7	120
21	Q	11-8	50	Q	11-8	121
22	R	11-9	51	R	11-9	122
23	S	0-2	22	S	0-2	123
24	T	0-3	23	T	0-3	124
25	U	0-4	24	U	0-4	125
26	V	0-5	25	V	0-5	126
27	W	0-6	26	W	0-6	127
30	X	0-7	27	X	0-7	130
31	Y	0-8	30	Y	0-8	131
32	Z	0-9	31	Z	0-9	132
33	0	0	12	0	0	060
34	1	1	01	1	1	061
35	2	2	02	2	2	062
36	3	3	03	3	3	063
37	4	4	04	4	4	064
40	5	5	05	5	5	065
41	6	6	06	6	6	066
42	7	7	07	7	7	067
43	8	8	10	8	8	070
44	9	9	11	9	9	071
45	+	12	60	+	12-8-6	053
46	-	11	40	-	11	055
47	*	11-8-4	54	*	11-8-4	052
50	/	0-1	21	/	0-1	057
51	(	0-8-4	34	(	12-8-5	050
52	)	12-8-4	74	)	11-8-5	051
53	$	11-8-3	53	$	11-8-3	044
54	=	8-3	13	=	8-6	075
55	blank	no punch	20	blank	no punch	040
56	, (comma)	0-8-3	33	, (comma)	0-8-3	054
57	. (period)	12-8-3	73	. (period)	12-8-3	056
60	≡	0-8-6	36	#	8-3	043
61	[	8-7	17	[	12-8-2	133
62	]	0-8-2	32	]	11-8-2	135
63	%[††]	8-6	16	%[††]	0-8-4	045
64	≠	8-4	14	" (quote)	8-7	042
65	↱	0-8-5	35	_ (underline)	0-8-5	137
66	∨	11-0 or 11-8-2[†††]	52	!	12-8-7 or 11-0[†††]	041
67	∧	0-8-7	37	&	12	046
70	↑	11-8-5	55	' (apostrophe)	8-5	047
71	↓	11-8-6	56	?	0-8-7	077
72	<	12-0 or 12-8-2[†††]	72	<	12-8-4 or 12-0[†††]	074
73	>	11-8-7	57	>	0-8-6	076
74	≤	8-5	15	@	8-4	100
75	≥	12-8-5	75	\	0-8-2	134
76	¬	12-8-6	76	− (circumflex)	11-8-7	136
77	; (semicolon)	12-8-7	77	; (semicolon)	11-8-6	073

[†] Twelve zero bits at the end of a 60-bit word in a zero byte record are an end of record mark rather than two colons.

[††] In installations using a 63-graphic set, display code 00 has no associated graphic or card code; display code 63 is the colon (8-2 punch). The % graphic and related card codes do not exist and translations yield a blank (55_8).

[†††] The alternate Hollerith (026) and ASCII (029) punches are accepted for input only.

TABLE A-2. AMERICAN NATIONAL STANDARD CODE FOR INFORMATION INTERCHANGE (ASCII) WITH PUNCHED CARD CODES AND EBCDIC TRANSLATION

	b8 b7 b6 b5	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
b4 b3 b2 b1	COL ROW	0	1	2	3	4	5	6	7	8	9	10 (A)	11 (B)	12 (C)	13 (D)	14 (E)	15 (F)
0 0 0 0	0	NUL 12-0-9-8-1 NUL 00	DLE 12-11-9-8-1 DLE 10	SP no-punch SP 40	0 0 0 F0	@ 8-4 @ 7C	P 11-7 P D7	` 8-1 ` 79	p 12-11-7 p 97	11-0-9-8-1 DS 20	12-11-0-9-8-1 30	12-0-9-1 41	12-11-9-8 58	12-11-0-9-6 76	12-11-8-7 9F	12-11-0-8 88	12-11-9-8-4 DC
0 0 0 1	1	SOH 12-9-1 SOH 01	DC1 11-9-1 DC1 11	! 12-8-7 | 4F	1 1 1 F1	A 12-1 A C1	Q 11-8 Q D8	a 12-0-1 a B1	q 12-11-8 q 98	0-9-1 SOS 21	9-1 31	12-0-9-2 42	11-8-1 59	12-11-0-9-7 77	11-0-8-1 A0	12-11-0-9 B9	12-11-9-8-5 DD
0 0 1 0	2	STX 12-9-2 STX 02	DC2 11-9-2 DC2 12	" 8-7 " 7F	2 2 2 F2	8 12-2 B C2	R 11-9 R D9	b 12-0-2 b 82	r 12-11-9 r 99	0-9-2 FS 22	11-9-8-2 CC 1A	12-0-9-3 43	11-0-9-2 62	12-11-0-9-8 78	11-0-8-2 AA	12-11-0-8-2 8A	12-11-9-8-6 DE
0 0 1 1	3	ETX 12-9-3 ETX 03	DC3 11-9-3 TM 13	# 8-3 # 78	3 3 3 F3	C 12-3 C C3	S 0-2 S E2	c 12-0-3 c 83	s 11-0-2 s A2	0-9-3 23	9-3 33	12-0-9-4 44	11-0-9-3 63	12-0-8-1 80	11-0-8-3 AB	12-11-0-8-3 B8	12-11-9-8-7 DF
0 1 0 0	4	EOT 9-7 EDT 37	DC4 9-8-4 DC4 3C	$ 11-8-3 $ 5B	4 4 4 F4	D 12-4 D C4	T 0-3 T E3	d 12-0-4 d 84	t 11-0-3 t A3	0-9-4 8YP 24	9-4 PN 34	12-0-9-5 45	11-0-9-4 64	12-0-8-2 8A	11-0-8-4 AC	12-11-0-8-4 8C	11-0-9-8-2 EA
0 1 0 1	5	ENQ 0-9-8-5 ENQ 2D	NAK 9-8-5 NAK 3D	% 0-8-4 % 6C	5 5 5 F5	E 12-5 E C5	U 0-4 U E4	e 12-0-5 e 85	u 11-0-4 u A4	11-9-5 NL 15	9-5 RS 35	12-0-9-6 46	11-0-9-5 65	12-0-8-3 88	11-0-8-5 AD	12-11-0-8-5 BD	11-0-9-8-3 EB
0 1 1 0	6	ACK 0-9-8-6 ACK 2E	SYN 9-2 SYN 32	& 12 & 50	6 6 6 F6	F 12-6 F C6	V 0-5 V E5	f 12-0-6 f 86	v 11-0-5 v A5	12-9-6 LC 06	9-6 UC 36	12-0-9-7 47	11-0-9-6 66	12-0-8-4 8C	11-0-8-6 AE	12-11-0-8-6 BE	11-0-9-8-4 ⊣ EC
0 1 1 1	7	8EL 0-9-8-7 8EL 2F	ETB 0-9-6 ET8 26	' 8-5 ' 7D	7 7 7 F7	G 12-7 G C7	W 0-6 W E6	g 12-0-7 g B7	w 11-0-6 w A6	11-9-7 IL 17	12-9-8 GE 08	12-0-9-8 48	11-0-9-7 67	12-0-8-5 8D	11-0-8-7 AF	12-11-0-8-7 8F	11-0-9-8-5 ED
1 0 0 0	8	BS 11-9-6 8S 16	CAN 11-9-8 CAN 18	(12-8-5 (4D	8 8 8 F8	H 12-8 H C8	X 0-7 X E7	h 12-0-8 h 88	x 11-0-7 x A7	0-9-8 28	9-8 38	12-8-1 49	11-0-9-8 68	12-0-8-6 8E	12-11-0-8-1 80	12-0-9-8-2 CA	11-0-9-8-6 EE
1 0 0 1	9	HT 12-9-5 HT 05	EM 11-9-8-1 EM 19	) 11-8-5) 5D	9 9 9 F9	I 12-9 I C9	Y 0-8 Y E8	i 12-0-9 i 89	y 11-0-8 y A8	0-9-8-1 29	9-8-1 39	12-11-9-1 51	0-8-1 89	12-0-8-7 8F	12-11-0-1 B1	12-0-9-8-3 CB	11-0-9-8-7 EF
1 0 1 0	10 (A)	LF 0-9-5 LF 25	SUB 9-8-7 SU8 3F	* 11-8-4 * 5C	: 8-2 : 7A	J 11-1 J D1	Z 0-9 Z E9	j 12-11-1 j 91	z 11-0-9 z A9	0-9-8-2 SM 2A	9-8-2 3A	12-11-9-2 52	12-11-0 70	12-11-8-1 90	12-11-0-2 82	12-0-9-8-4 ┘ CC	12-11-0-9-8-2 I (LVM) FA
1 0 1 1	11 (8)	VT 12-9-8-3 VT 0B	ESC 0-9-7 ESC 27	+ 12-8-6 + 4E	; 11-8-6 ; 5E	K 11-2 K D2	[12-8-2 ¢ 4A	k 12-11-2 k 92	{ 12-0 { C0	0-9-8-3 CU2 28	9-8-3 CU3 38	12-11-9-3 53	12-11-0-9-1 71	12-11-8-2 9A	12-11-0-3 B3	12-0-9-8-5 CD	12-11-0-9-8-3 F8
1 1 0 0	12 (C)	FF 12-9-8-4 FF 0C	FS 11-9-8-4 IFS 1C	, 0-8-3 , 68	< 12-8-4 < 4C	L 11-3 L D3	\ 0-B-2 \ E0	l 12-11-3 l 93	¦ 12-11 ¦ 6A	0-9-8-4 2C	12-9-4 PF 04	12-11-9-4 54	12-11-0-9-2 72	12-11-8-3 98	12-11-0-4 84	12-0-9-8-6 Ч CE	12-11-0-9-8-4 FC
1 1 0 1	13 (D)	CR 12-9-8-5 CR 0D	GS 11-9-8-5 IGS 1D	— 11 — 60	= 8-6 = 7E	M 11-4 MD4	] 11-8-2 ! 5A	m 12-11-4 m 94	} 11-0 } D0	12-9-8-1 RLF 09	11-9-4 RES 14	12-11-9-5 55	12-11-0-9-3 73	12-11-8-4 9C	12-11-0-5 85	12-0-9-8-7 CF	12-11-0-9-8-5 FD
1 1 1 0	14 (E)	SO 12-9-B-6 SO 0E	RS 11-9-8-6 IRS 1E	. 12-8-3 . 48	> 0-8-6 > 6E	N 11-5 N D5	^ 11-8-7 ¬ 5F	n 12-11-5 n 95	~ 11-0-1 ~ A1	12-9-8-2 SMM 0A	9-8-6 3E	12-11-9-6 56	12-11-0-9-4 74	12-11-8-5 9D	12-11-0-6 86	12-11-9-8-2 DA	12-11-0-9-8-6 FE
1 1 1 1	15 (F)	SI 12-9-8-7 SI 0F	US 11-9-8-7 IUS 1F	/ 0-1 / 61	? 0-8-7 ? 6F	O 11-6 D D6	0̄-8-5 _ 6D	o 12-11-6 o 96	DEL 12-9-7 DEL 07	11-9-8-3 CU1 18	11-0-9-1 E1	12-11-9-7 57	12-11-0-9-5 75	12-11-8-6 9E	12-11-0-7 87	12-11-9-8-3 DB	EO 12-11-0-9-8-7 FF

LEGEND

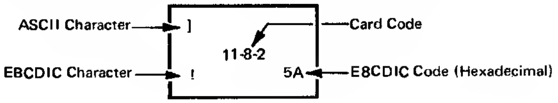


TABLE A-3. EXTENDED BINARY CODED DECIMAL INTERCHANGE CODE (EBCDIC) WITH PUNCHED CARD CODES AND ASCII TRANSLATION

	BITS 0 1 2 3	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
BITS 4 5 6 7	1ST HEX 2ND	0	1	2	3	4	5	6	7	8	9	A (10)	B (11)	C (12)	D (13)	E (14)	F (15)
0 0 0 0	0	NUL 12-0-9-8-1 NUL 00	DLE 12-11-9-8-1 DLE 10	DS 11-0-9-8-1 80	12-11-0-9-8-1 90	SP no punch SP 20	& 12 & 26	- 11 - 2D	12-11-0 BA	12-0-8-1 C3	12-11-8-1 CA	11-0-8-1 D1	12-11-0-8-1 D8	{ 12-0 { 7B	} 11-0 } 7D	\ 0-8-2 \ 5C	0 0 0 30
0 0 0 1	1	SDH 12-9-1 SDH 01	DC1 11-9-1 DC1 11	SDS 0-9-1 81	9-1 91	12-0-9-1 A0	12-11-9-1 A9	/ 0-1 / 2F	12-11-0-9-1 BB	a 12-0-1 a 61	j 12-11-1 j 6A	~ 11-0-1 ~ 7E	12-11-0-1 D9	A 12-1 A 41	J 11-1 J 4A	11-0-9-1 9F	1 1 1 31
0 0 1 0	2	STX . 9-2 STX 02	DC2 11-9-2 DC2 12	FS 0-9-2 B2	SYN 9-2 SYN 16	12-0-9-2 A1	12-11-9-2 AA	11-0-9-2 82	12-11-0-9-2 BC	b 12-0-2 b 62	k 12-11-2 k 6B	s 11-0-2 s 73	12-11-0-2 DA	B 12-2 B 42	K 11-2 K 4B	S 0-2 S 53	2 2 2 32
0 0 1 1	3	ETX 12-9-3 ETX 03	TM 11-9-3 DC3 13	0-9-3 B3	9-3 93	12-0-9-3 A2	12-11-9-3 AB	11-0-9-3 B3	12-11-0-9-3 BD	c 12-0-3 c 63	l 12-11-3 l 6C	t 11-0-3 t 74	12-11-0-3 DB	C 12-3 C 43	L 11-3 L 4C	T 0-3 T 54	3 3 3 33
0 1 0 0	4	PF 12-9-4 9C	RES 11-9-4 9D	BYP 0-9-4 B4	PN 9-4 94	12-0-9-4 A3	12-11-9-4 AC	11-0-9-4 B4	12-11-0-9-4 BE	d 12-0-4 d 64	m 12-11-4 m 6D	u 11-0-4 u 75	12-11-0-4 DC	D 12-4 D 44	M 11-4 M 4D	U 0-4 U 55	4 4 4 34
0 1 0 1	5	HT 12-8-5 HT 09	NL 11-9-5 B5	LF 0-9-5 LF 0A	RS 9-5 95	12-0-9-5 A4	12-11-9-5 AD	11-0-9-5 B5	12-11-0-9-5 BF	e 12-0-5 e 65	n 12-11-5 n 6E	v 11-0-5 v 76	12-11-0-5 DD	E 12-5 E 45	N 11-5 N 4E	V 0-5 V 56	5 5 5 35
0 1 1 0	6	LC 12-9-6 B6	BS 11-9-6 8S 08	ETB 0-9-6 ETB 17	UC 9-6 98	12-0-9-6 A5	12-11-9-6 AE	11-0-9-8 B6	12-11-0-9-6 C0	f 12-0-6 f 66	o 12-11-8 o 6F	w 11-0-8 w 77	12-11-0-6 DE	F 12-8 F 46	O 11-8 O 4F	W 0-6 W 57	B 6 B 36
0 1 1 1	7	DEL 12-9-7 DEL 7F	IL 11-9-7 B7	ESC 0-9-7 ESC 1B	EDT 9-7 EDT 04	12-0-9-7 A6	12-11-9-7 AF	11-0-9-7 B7	12-11-0-9-7 C1	g 12-0-7 g 67	p 12-11-7 p 70	x 11-0-7 x 78	12-11-0-7 DF	G 12-7 G 47	P 11-7 P 50	X 0-7 X 58	7 7 7 37
1 0 0 0	B	GE 12-9-8 97	CAN 11-9-8 CAN 1B	0-9-8 88	9-8 98	12-0-9-8 A7	12-11-9-8 B0	11-0-9-8 BB	12-11-0-9-8 C2	h 12-0-8 h 68	q 12-11-8 q 71	y 11-0-8 y 79	12-11-0-8 E0	H 12-8 H 48	Q 11-8 Q 51	Y 0-8 Y 59	B B B 38
1 0 0 1	9	RLF 12-9-8-1 BD	EM 11-9-8-1 EM 19	0-9-8-1 89	9-8-1 99	12-8-1 A8	11-8-1 B1	0-8-1 B9	` 8-1 ` 60	i 12-0-8 i 69	r 12-11-9 r 72	z 11-0-9 z 7A	12-11-0-9 E1	I 12-9 I 49	R 11-8 R 52	Z 0-9 Z 5A	9 9 9 39
1 0 1 0	A (10)	SMM 12-9-8-2 8E	CC 11-9-8-2 92	SM 0-9-8-2 8A	9-8-2 9A	¢ 12-8-2 [5B	! 11-8-2] 5D	¦ 12-11 ¦ 7C	: 8-2 : 3A	12-0-8-2 C4	12-11-8-2 CB	11-0-8-2 D2	12-11-0-8-2 E2	12-0-9-8-2 E8	12-11-8-8-2 EE	11-0-9-8-2 F4	I (LVM) 12-11-0-9-8-2 FA
1 0 1 1	B (11)	VT 12-9-8-3 VT 0B	CU1 11-9-8-3 BF	CU2 0-9-8-3 8B	CU3 9-8-3 9B	. 12-8-3 . 2E	$ 11-8-3 $ 24	, 0-8-3 , 2C	# B-3 # 23	12-0-8-3 C5	12-11-8-3 CC	11-0-8-3 D3	12-11-0-8-3 E3	12-0-9-8-3 E9	12-11-9-8-3 EF	11-0-9-8-3 F6	12-11-0-9-8-3 FB
1 1 0 0	C (12)	FF 12-9-8-4 FF 0C	IFS 11-9-8-4 FS 1C	0-9-8-4 8C	DC4 9-8-4 DC4 14	< 12-8-4 < 3C	* 11-8-4 * 2A	% 0-8-4 % 25	@ 8-4 @ 40	12-0-8-4 C6	12-11-8-4 CD	11-0-8-4 D4	12-11-0-8-4 E4	⑀ 12-0-9-8-4 EA	12-11-9-8-4 F0	⑁ 11-0-9-8-4 F8	12-11-0-9-8-4 FC
1 1 0 1	D (13)	CR 12-9-8-5 CR 0D	IGS 11-9-8-5 GS 1D	ENQ 0-9-8-5 ENQ 05	NAK 9-8-5 NAK 15	(12-8-5 (28	) 11-8-5) 29	_ 0-8-6 _ 5F	' 8-5 ' 27	12-0-8-5 C7	12-11-6-5 CE	11-0-8-5 D5	12-11-0-8-5 E5	12-0-8-8-5 EB	12-11-9-8-5 F1	11-0-9-8-5 F7	12-11-0-9-8-5 FD
1 1 1 0	E (14)	SD 12-9-8-6 SO 0E	IRS 11-9-8-6 RS 1E	ACK 0-9-8-8 ACK 06	9-8-8 9E	+ 12-8-6 + 2B	; 11-8-6 ; 3B	> 0-8-6 > 3E	= 8-6 = 3D	12-0-8-6 C8	12-11-8-6 CF	11-0-8-6 D6	12-11-0-8-8 E6	⑂ 12-0-9-8-6 EC	12-11-9-8-8 F2	11-0-8-8-6 F8	12-11-0-9-8-6 FE
1 1 1 1	F (15)	SI 12-9-8-7 SI 0F	IUS 11-9-8-7 US 1F	8EL 0-9-8-7 8EL 07	SUB 9-8-7 SUB 1A	| 12-8-7 ! 21	¬ 11-8-7 ^ 5E	? 0-8-7 ? 3F	" 8-7 " 22	12-0-8-7 C9	12-11-8-7 D0	11-0-8-7 D7	12-11-0-8-7 E7	12-0-8-8-7 ED	12-11-9-8-7 F3	11-0-9-8-7 F9	12-11-0-9-8-7 ED FF

LEGEND

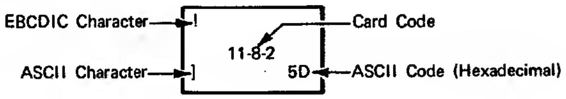


TABLE A-4. CONTROL DATA CHARACTER SETS SHOWING TRANSLATIONS BETWEEN DISPLAY CODE AND ASCII/EBCDIC

Display Code		ASCII				EBCDIC				Display Code		ASCII				EBCDIC			
		Upper-case		Lower-case		Upper-case		Lower-case				Upper-case		Lower-case		Upper-case		Lower-case	
Octal	Char.	Char.	Hex	Char.	Hex	Char.	Hex	Char.	Hex	Octal	Char.	Char.	Hex	Char.	Hex	Char.	Hex	Char.	Hex
00	:	:	3A	SUB	1A	:	7A	SUB	3F	40	5	5	35	NAK	15	5	F5	NAK	3D
01	A	A	41	a	61	A	C1	a	81	41	6	6	36	SYN	16	6	F6	SYN	32
02	B	B	42	b	62	B	C2	b	82	42	7	7	37	ETB	17	7	F7	ETB	26
03	C	C	43	c	63	C	C3	c	83	43	8	8	38	CAN	18	8	F8	CAN	18
04	D	D	44	d	64	D	C4	d	84	44	9	9	39	EM	19	9	F9	EM	19
05	E	E	45	e	65	E	C5	e	85	45	+	+	2B	VT	0B	+	4E	VT	0B
06	F	F	46	f	66	F	C6	f	86	46	–	–	2D	CR	0D	–	60	CR	0D
07	G	G	47	g	67	G	C7	g	87	47	*	*	2A	LF	0A	*	5C	LF	25
10	H	H	4B	h	68	H	C8	h	88	50	/	/	2F	SI	0F	/	61	SI	0F
11	I	I	49	i	69	I	C9	i	89	51	(	(	28	BS	08	(	4D	BS	16
12	J	J	4A	j	6A	J	D1	j	91	52	)	)	29	HT	09	)	5D	HT	05
13	K	K	4B	k	6B	K	D2	k	92	53	$	$	24	EOT	04	$	5B	EOT	37
14	L	L	4C	l	6C	L	D3	l	93	54	=	=	3D	GS	1D	=	7E	IGS	1D
15	M	M	4D	m	6D	M	D4	m	94	55	SP	SP	20	NUL	00	SP	40	NUL	00
16	N	N	4E	n	6E	N	D5	n	95	56	,	,	2C	FF	0C	,	6B	FF	0C
17	O	O	4F	o	6F	O	D6	o	96	57	.	.	2E	SO	0E	.	48	SO	0E
20	P	P	50	p	70	P	D7	p	97	60	≡ #	#	23	ETX	03	#	7B	ETX	03
21	Q	Q	51	q	71	Q	D8	q	98	61	[	[	5B	FS	1C	¢	4A	IFS	1C
22	R	R	52	r	72	R	D9	r	99	62	]	]	5D	SOH	01	!	5A	SOH	01
23	S	S	53	s	73	S	E2	s	A2	63	%	%	25	ENQ	05	%	6C	ENQ	2D
24	T	T	54	t	74	T	E3	t	A3	64	≠ "	"	22	STX	02	"	7F	STX	02
25	U	U	55	u	75	U	E4	u	A4	65	→ _	_	5F	DEL	7F	_	6D	DEL	07
26	V	V	56	v	76	V	E5	v	A5	66	∨ !	!	21	}	7D	|	4F	}	D0
27	W	W	57	w	77	W	E6	w	A6	67	∧ &	&	26	ACK	06	&	50	ACK	2E
30	X	X	58	x	78	X	E7	x	A7	70	↑ '	'	27	BEL	07	'	7D	BEL	2F
31	Y	Y	59	y	79	Y	E8	y	A8	71	↓ ?	?	3F	US	1F	?	6F	IUS	1F
32	Z	Z	5A	z	7A	Z	E9	z	A9	72	<	<	3C	{	7B	<	4C	{	C0
33	0	0	30	DLE	10	0	F0	DLE	10	73	>	>	3E	RS	1E	>	6E	IRS	1E
34	1	1	31	DC1	11	1	F1	DC1	11	74	≤ @	@	40	'	60	@	7C	'	79
35	2	2	32	DC2	12	2	F2	DC2	12	75	≥ \	\	5C	¦	7C	\	E0	¦	6A
36	3	3	33	DC3	13	3	F3	TM	13	76	¬ ^	^	5E	~	7E	¬	5F	~	A1
37	4	4	34	DC4	14	4	F4	DC4	3C	77	;	;	3B	ESC	1B	;	5E	ESC	27

NOTES:

1. The terms uppercase and lowercase apply only to the case conversions, and do not necessarily reflect any true case.
2. When translating from Display Code to ASCII/EBCDIC, the uppercase equivalent character is taken.
3. When translating from ASCII/EBCDIC to Display Code, the uppercase and lowercase characters fold together to a single Display Code equivalent character.
4. All ASCII and EBCDIC codes not listed are translated to Display Code 55_8(SP).
5. Where two Display Code graphics are shown for a single octal code, the leftmost graphic corresponds to the CDC 64-character set, and the rightmost graphic corresponds to the CDC 64-character ASCII subset.
6. In a 63-character set system, the display code for the : graphic is 63. The % character does not exist, and translations from ASCII/EBCDIC % or ENQ yield blank (55_8).

TABLE A-5. FULL ASCII CHARACTER SET

← 128-Character Set →

← 96-Character Subset →

← 64-Character Subset →

b7 →					0	0	0	0	1	1	1	1
b6 →					0	0	1	1	0	0	1	1
b5 →					0	1	0	1	0	1	0	1
Bits b4 ↓	b3 ↓	b2 ↓	b1 ↓	ROW ↓ COLUMN →	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	’	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	¦
1	1	0	1	D	CR	GS	—	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	^	n	~
1	1	1	1	F	SI	US	/	?	O	___	o	DEL

LIST OF KEYWORDS B

1. The following general field names are applicable to all supervisory messages:

PFCSFC	Primary and secondary function code
PFC	Primary function code
EB	Error bit
RB	Response bit
SFC	Secondary function
RC	Reason code
SPMSG0	Word 0 of any supervisory message
SPMSG1	Word 1 of any supervisory message
⋮	⋮
SPMSG9	Word 9 of any supervisory message

2. The following fields are in connection management messages:

CONACN	Application connection number
CONABL	Application block limit
CONHW	Hardwired line
CONOT	Device type
CONORD	Device ordinal
CONTNM	Terminal name
CONANM	Name of requesting application
CONPW	Page width
CONPL	Page length
CONOWT	Controlling user's terminal name
CONPAR	First word of parameters passed on a CON/REQ
CONACT	Application input character type
CONALN	Application list number

3. The following fields are in other messages:

FCACN	ACN in all FC messages
FCABN	ABN in FC/ACK
LSTACN	ACN in all LST messages
LSTALN	ALN in LST/SWH
CTRSTR	String in CTRL/OEF
DCACN	ACN in DC/CICT
ERRMSG	Message text in ERR/LGL
ERRABH	ABH in ERR/LGL
MSGNCH	NCHAR in MSG/LOP
MSGTEX	TEXT in MSG/LOP
SHUTYP	Shut down type in SHUT/INSO

4. The following fields in the application block header are for supervisory or nonsupervisory messages:

ABHABT	Application block type
ABHAOR	Addressing information
ABHABN	Application block number
ABHACT	Application character type
ABHIBU	Input block undeliverable flag
ABHNFE	No format effectors flag
ABHXPT	Transparent bit
ABHCAN	Cancel bit or punch-banner-card
ABHBIT	Parity error or auto-input flag
ABHTLC	Text length in characters
ABHWORD	Whole word of ABH

5. Control Oata defined values of the following symbols are available to the application. Release values are indicated in parentheses.

CON	PFC for CON messages (63_{16})
LCONRQ	Length of CON/REQ message not including APARAM (4)
LCORQR	Length of CON/REQ response (1)
REQ	SFC for CON/REQ (00)
CONREQ	PFC and SFC for CON/REQ (6300_{16})
LCONAC	Length of CON/ACRQ (2)
ACRQ	SFC for CON/ACRQ (02_{16})
CONACR	PFC and SFC for CON/ACRQ (6302_{16})
LCONEN	Length of CON/ENO (2)
ENOO	SFC for CON/ENO (06_{16})
CONENO	PFC and SFC for CON/ENO (6306_{16})

CB	SFC for CON/CB (05$_{16}$)
CONCB	PFCSFC for CONCB (6305$_{16}$)
LCONCB	Length of the CON/CB message
LCTRL	Length of all the CTRL messages
CTRL	PFC for CTRL messages (C1$_{16}$)
START	SFC for CTRL/START (05$_{16}$)
CTRSTR	PFC and SFC for CTRL/START (C105$_{16}$)
STOPP	SFC for CTRL/STOP (06$_{16}$)
CTRSTP	PFC and SFC for CTRL/STOP (C106$_{16}$)
STPD	SFC for CTRL/STPD
CTRSTD	PFC/SFC for CTRL/STPD
DEFF	SFC for CTRL/DEF (C104$_{16}$)
CTRDEF	PFC/SFC for CTRL/DEF (C104$_{16}$)
LDC	Length of DC/CICT (1)
DC	PFC for DC/CICT (C2$_{16}$)
CICT	SFC for DC/CICT (00$_{16}$)
DCCICT	PFCSFC for DC/CICT (C200$_{16}$)
LLST	Length of LST messages (1)
LST	PFC for LST messages (C0$_{16}$)
OFF	SFC for LST/OFF (1)
LSTOFF	PFC and SFC for LST/OFF (C001$_{16}$)
ON	SFC for LST/ON (0)
LSTON	PFC and SFC for LST/ON (C000$_{16}$)
SWH	SFC for LST/SWH (02$_{16}$)
LSTSWH	PFC and SFC for LST/SWH (C002$_{16}$)
LMSG	Length of MSG messages (1)
MSG	PFC for MSG (E0$_{16}$)
LOP	SFC for MSG/LOP (07$_{16}$)
MSGLOP	PFC and SFC for MSG/LOP (E007$_{16}$)
LSHUT	Length of SHUT messages (1)
SHUT	PFC for SHUT messages (42$_{16}$)
INSD	SFC for SHUT/INSD (06$_{16}$)
SHUINS	PFC and SFC for SHUT/INSD (4206$_{16}$)
LTCH	Length of TCH messages (1)
TCH	PFC for TCH (64$_{16}$)
TCHAR	SFC for TCH/TCHAR (00$_{16}$)
TCHTCH	PFC and SFC for TCH/TCHAR (6400$_{16}$)

SUPERVISORY MESSAGES

NAM TO THE APPLICATION

PFC/SFC	Hex Code	Meaning
CON/REQ	4206	Request logical connection
CON/CB	6305	Connection broken
FC/BRK	8300	Break
FC/STP	8305	Suspend data traffic
FC/STRT	8306	Resume data traffic
FC/INIT	8307	Logical connection initialized
FC/ACK	8302	Block delivered
FC/NAK	8303	Block not delivered
FC/INACT	8304	Connection inactive
ERR/LGL	8401	Logical error
SHUT/INSD	4206	Network shutdown
TCH/TCHAR	6400	Terminal characteristics changed

APPLICATION TO NAM

PFC/SFC	Hex Code	Meaning
CON/ACRQ	6302	Application connection request
CON/END	6306	End connection
MSG/LOP	E007	Message to local operator
DC/CICT	C200	Change input character type
FC/RST	8301	Reset
CTRL/DEF[†]	C104	Define terminal characteristics
LST/OFF	C000	Temporary OFF
LST/ON	C001	Temporary ON
LST/SWH	C002	Switch lists

[†] Synchronous SM; all other SMs are asynchronous.

CONFIGURATION FILES D

Two files are used by the network host products software in establishing, initiating, and operating the network:

- The network configuration file (NCF)
- The local configuration file (LCF)

Both files are direct access, random indexed permanent files.

The NCF contains information that describes the physical and logical configuration of the network elements, such as:

- The host and its connected couplers
- The NPUs that are part of the network
- The physical and logical connections for the host, couplers, and NPUs

The LCF describes those components of the network which belong to a host computer, such as:

- The application program names known to the network
- The lines between NPUs and terminals
- The initial terminal characteristics

These components are described by the Network Definition Language (NDL) statements, and are created by the NDL Processor. The NDL Processor executes as a batch job separately from the network environment. For further information, refer to the NDL reference manual.

INDEX

COMMENT SHEET

CONTROL DATA
CORPORATION

TITLE: Network Access Method Version 1, FORTRAN Application Programmer's System Bulletin

PUBLICATION NO. 60480400 REVISION A

This form is not intended to be used as an order blank. Control Data Corporation solicits your comments about this manual with a view to improving its usefulness in later editions.

Applications for which you use this manual.

Do you find it adequate for your purpose?

What improvements to this manual do you recommend to better serve your purpose?

Note specific errors discovered (please include page number reference).

General comments:

FROM NAME: ______________________ POSITION: ______________________

COMPANY
NAME: ____

ADDRESS: ____

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE STAPLE

FOLD FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Publications and Graphics Division
215 Moffett Park Drive
Sunnyvale, California 94086

FOLD FOLD

STAPLE STAPLE